AIM :

To analyze the working of Google's Deep Learning powerful AI i.e. AlphaGo and discovering its implementations. Also, study various latest implementations and derive a hybrid model named as ***'MinAlphaGo'*** *with a better and faster user interface along with the analysis of game state at any instant.*

OBJECTIVE:

- Studying AlphaGo (Google's Deep Learning powerful AI).
- Study of Google's research paper titled as : 1.Mastering the game of Go with deep neural networks and tree search : https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf
- Combining plus points of BetaGo and Leela Game Engine implementations and coming out with hybrid version named as 'MinAlphaGo'.
- Comparing the interface of Leela and MinAlphaGo.

INTRODUCTION:

The ancient Chinese game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. Despite its relatively simple rules, Go is very complex, even more so than chess, and possesses more possibilities than the total number of atoms in the visible universe.
AlphaGo, an AI computer program which was developed by Alphabet Inc.'s Google DeepMind in London in October 2015, became the first

Computer Go program to beat a human professional Go player on a official-sized 19×19 board.

AlphaGo's underlying algorithms are potentially more general-purpose, and may be evidence that the scientific community is making progress towards artificial general intelligence.
Hence we intend to analyze and research about the AlphaGo AI and figure out its applications in Space Exploration or in Medical Field.

In chess, each player begins with 16 pieces of six different types. Each piece type moves differently. The goal of the game is to capture the opponent's king. Go starts with an empty board. At each turn, a player places a stone (the equivalent of a piece in chess) on the board. Stones all obey the same rules. The goal of the game is to capture as much territory as possible. It can therefore be argued that **Go has simpler rules than chess.**

In spite of the fact that the rules of Go might appear simpler than the rules of chess, **the complexity of Go is higher. At each game state, a player is faced with a choice of a greater number of possible moves compared to chess (about 250 in Go vs. 35 in chess).**

Because of this, the total number of possible games of Go has been estimated at 10^{761}, compared to 10^{120} for chess. By combining deep learning and reinforcement learning in a series of artificial neural networks, **AlphaGo first learned human expert-level play in Go from 30 million moves from human games**.

Few major teams are trying to come out with the improvised versions of AI for the game of Go, based on the theory presented in the paper. We have come out with a hybrid model named as 'MinAlphaGo' , which encaptures the plus points of the two already existing implementations.

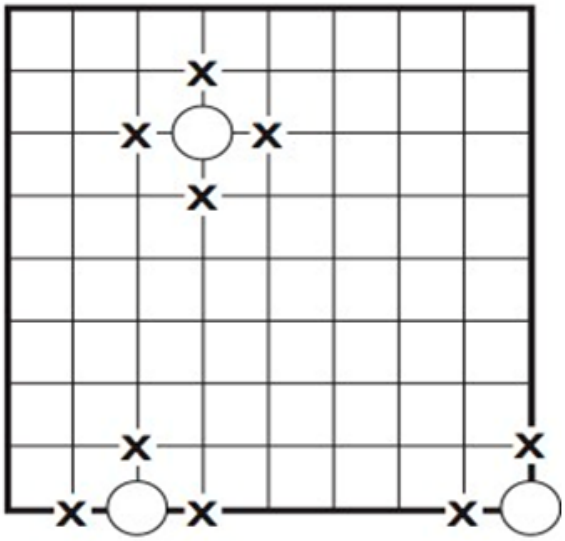
Scope of the Project:

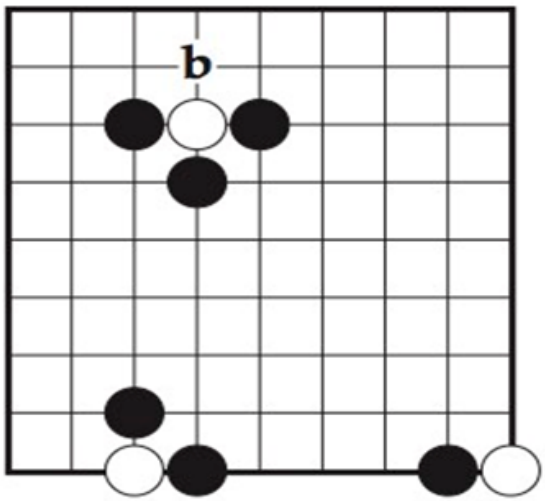
- This kind Of AI's act as Litmus Test for measuring the efficiency and power of other existing AI's.
- Game Engines serve as a good area for experimentation for any kind of AI, as it involves a lot of human intuitive capabilities.
- As AlphaGo can act as a general purpose algorithm so it can be used to solve problems of other domains like :
    - Climatic Modelling for Monitoring Environmental issues.
    - Medical images to make diagnoses or treatment plans of complex diseases (Disease Analysis).
    - Space Exploration.
    - Security.

RULES OF GO GAME –

- Although the official size of a Go board is 19 x 19 ,but it can be played on a 13 x 13 board and 9 x 9 board.
- The Objective of Go game is to capture more territories than the opponent and other players stone.
- A basic principle of Go is that a group of stones must have at least one liberty(open point bordering the group) to remain on the board.
- An enclosed liberty (or liberties) is called an eye, and a group of stones with two or more eyes is said to be unconditionally alive.
- Such groups cannot be captured, even if surrounded.
- A group with one eye or no eyes is "dead" and cannot resist eventual capture

- (the "ko rule") states that the stones on the board must never repeat a previous position of stones. Moves which would do so are forbidden, and thus only moves elsewhere on the board are permitted that turn





TOOLS & TECHNIQUES:

- Platform Used - Linux, Mach OSX
- Python 2.7.14
- Keras with TensorFlow Backend.
- Leela (Strong Go playing program or Game Engine)
- Yaml and Json.
- Flask(Python's Web Microframework).

IMPORTANT CONCEPTS:

Alphago is made up of number of relatively standard techniques and Networks:

- Convolution Neural Network
- Supervised Learning
- Reinforcement learning
- Monte Carlo Tree Search
- Value Network
- Policy Network and Fast Policy Network

CONVOLUTION NEURAL NETWORK :

A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers (often with a subsampling step) and then followed by one or more fully connected layers as in a standard multilayer neural network. The architecture of a CNN is designed to take advantage of the 2D structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units. In this article we will discuss the architecture of a CNN and the back propagation algorithm to compute the gradient with respect to the parameters of the model in order to use gradient based optimization. See the respective tutorials on convolution and pooling for more details on those specific operations.

The size of the Feature Map (Convolved Feature) is controlled by three parameters ,that we need to decide before the convolution step is performed:

Depth: Depth corresponds to the number of filters we use for the convolution operation. In the network .we are performing convolution of the original boat image using three distinct filters, thus producing three different feature maps as shown. You can think of these three feature maps as stacked 2d matrices, so, the 'depth' of the feature map would be three.

Stride: Stride is the number of pixels by which we slide our filter matrix over the input matrix. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2, then the filters jump 2 pixels at a time as we slide them around. Having a larger stride will produce smaller feature maps.

Zero-padding: Sometimes, it is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix. A nice feature of zero padding is that it allows us to control the size of the feature maps. Adding zero-padding is also called wide convolution, and not using zero-padding would be a narrow convolution.

SUPERVISED LEARNING -

Supervised learning is the machine learning task of inferring a function from labeled training data.[1] The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

REINFORCEMENT LEARNING -

Reinforcement learning (RL) is an area of machine learning inspired by behaviourist psychology, concerned with how software agentsought to take actions in an environment so as to maximize some notion of cumulative reward. The problem, due to its generality, is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms. In the operations research and control literature, the field where reinforcement learning methods are studied is called approximate dynamic programming. The problem has been studied in the theory of optimal control, though most studies are concerned with the existence of optimal solutions and their characterization, and not with the learning or approximation aspects. In economics and game theory, reinforcement learning may be used to explain how equilibrium may arise under bounded rationality.

POLICY NETWORK-

A policy network was trained on 30 million positions from games played by human experts, available at the KGS Go server. An accuracy on a withheld test-set of 57% was achieved. When I first read the paper, I was very surprised that this was possible. I would have thought that predicting human game moves is a problem that is too difficult to be solved with convolutional networks. The fact that a convolutional network was able to predict human game play with such high accuracy seems to suggest that a lot of human Go-play is rather intuitive, instead of deeply reflective.

A smaller policy network is trained as well. Its accuracy is much lower (24.2%), but is much faster (2 microseconds instead of 3 milliseconds: 1500 times faster).

VALUE NETWORK-

Then, a value network was trained on 30 million game positions obtained while the policy network played against itself. As a reminder, the value network is supposed to predict the likelihood of a win, give the current game-state. It is therefore similar to an evaluation function, with the difference that the value network is learned instead of designed.

We said that it is difficult to design evaluation functions that perform well at the beginning of a game, but that it becomes easier the closer the game approaches an end. The same effect is observed with the value network: the value network makes random predictions at the very beginning of the game, but becomes better at predicting the final game outcome the more moves have been played. The fact that both human-designed evaluation functions and learned value networks display the tendency to perform better towards the end of the game indicates that this tendency is not due to human limitations, but that it is something more fundamental.

MCTS

DeepMind's version of MCTS effectively reduces the depth and breadth of tree search by integrating convnets into AlphaGo: evaluating positions using a value network, and sampling actions using a policy network.

The tree is traversed by two-phase simulations: in-tree phase of each simulation begins at the root of the search tree and finishes when the simulation reaches a leaf node in L time steps, and second rollout phase of each simulation begins at leaf node and continues until the end of the game.

LEELA

- Leela is a strong Go playing program combining advances in Go programming and further original research into a small, easy to use graphical interface.
- Strong Go engine including support for multiple processors and GPU acceleration.
- Strength over 9 dan on 19 x 19, depending on hardware.
- Featuring Deep Learning technology.
- Easy to use graphical interface.
- Adjustable board size (up to 25x25!), playing levels, handicap and komi
- Fixed strength and time based difficulty levels
- Chinese rules with positional superko
- SGF format loading and saving
- Rated game mode with auto-adjusting difficulty levels
- Analysis features including critical variations for each move and winning odds.
- Graphical display of critical moves, territory, best lines, …

YAML:

YAML (YAML Ain't Markup Language) is a human-readable data serialization language. It is commonly used for configuration files, but could be used in many applications where data is being stored (e.g. debugging output) or transmitted (e.g. document headers). YAML targets many of the same communications applications as XML, but has taken a more minimal approach which intentionally breaks compatibility with SGML. YAML 1.2 is a superset of JSON , another minimalist data serialization format where braces and brackets are used instead of indentation. It can be incorporated into many different programming languages using supporting YAML libraries, including C/C++, Ruby, Python, Java, Perl, C#, PHP, and others.

FLASK :

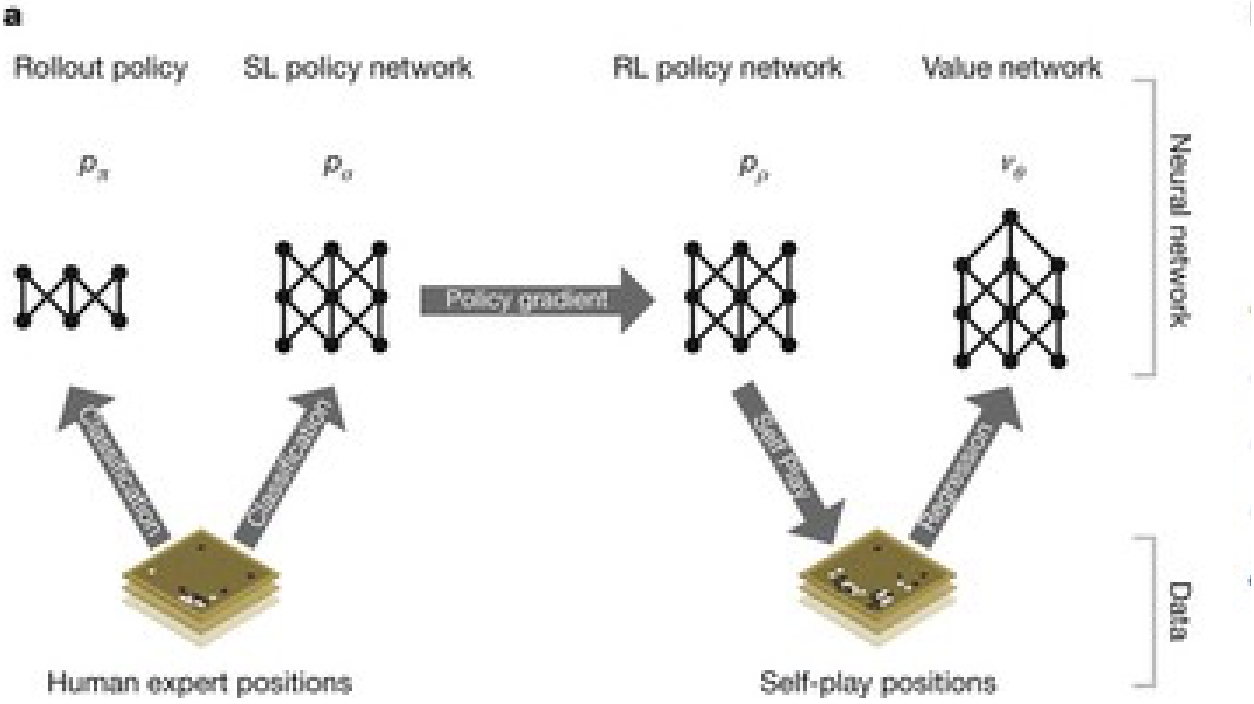
Flask is a Python web framework built with a small core and easy-to-extend philosophy.

Flask is a web application framework written in Python. It is developed by Armin Ronacher, who leads an international group of Python enthusiasts named Pocco. Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

SUMMARY OF RESEARCH PAPER:-

- New approach: uses **'Value Networks'** to evaluate board positions and **'Policy Networks'** to select moves.
- Combination of **Supervised Learning and Reinforcement Learning**
- New search algo: combines Monte Carlo simulation with value and policy networks (lookahead search)
- Using this search algo, **AlphaGo achieved 99.8% winning rate.**
- AlphaGo uses a SL policy to initialize the learning of an RL policy that gets perfected with self play, which they then estimate a value function from, which then plugs into MCTS that uses the SL policy to sample rollouts.
- **Optimal Function** : determines the outcomes of the game from every board position or state s.
- This optimal value function can be solved by traversing a search tree containing approx. $b \wedge d$ sequence.
  - Where b-game's breadth (no. of legal moves per position)
  - d-depth (game length)
- AlphaGo combines Monte Carlo simulations with value and policy networks called **"Asynchronous policy and value MCTS" (APV-MCTS).**
- **Training Pipeline -**

- ○ **Supervised Learning of policy network**
- ○ **Reinforcement Learning of policy network**
- ○ **Reinforcement Learning of value network**



Our Contribution :

Latest Works :

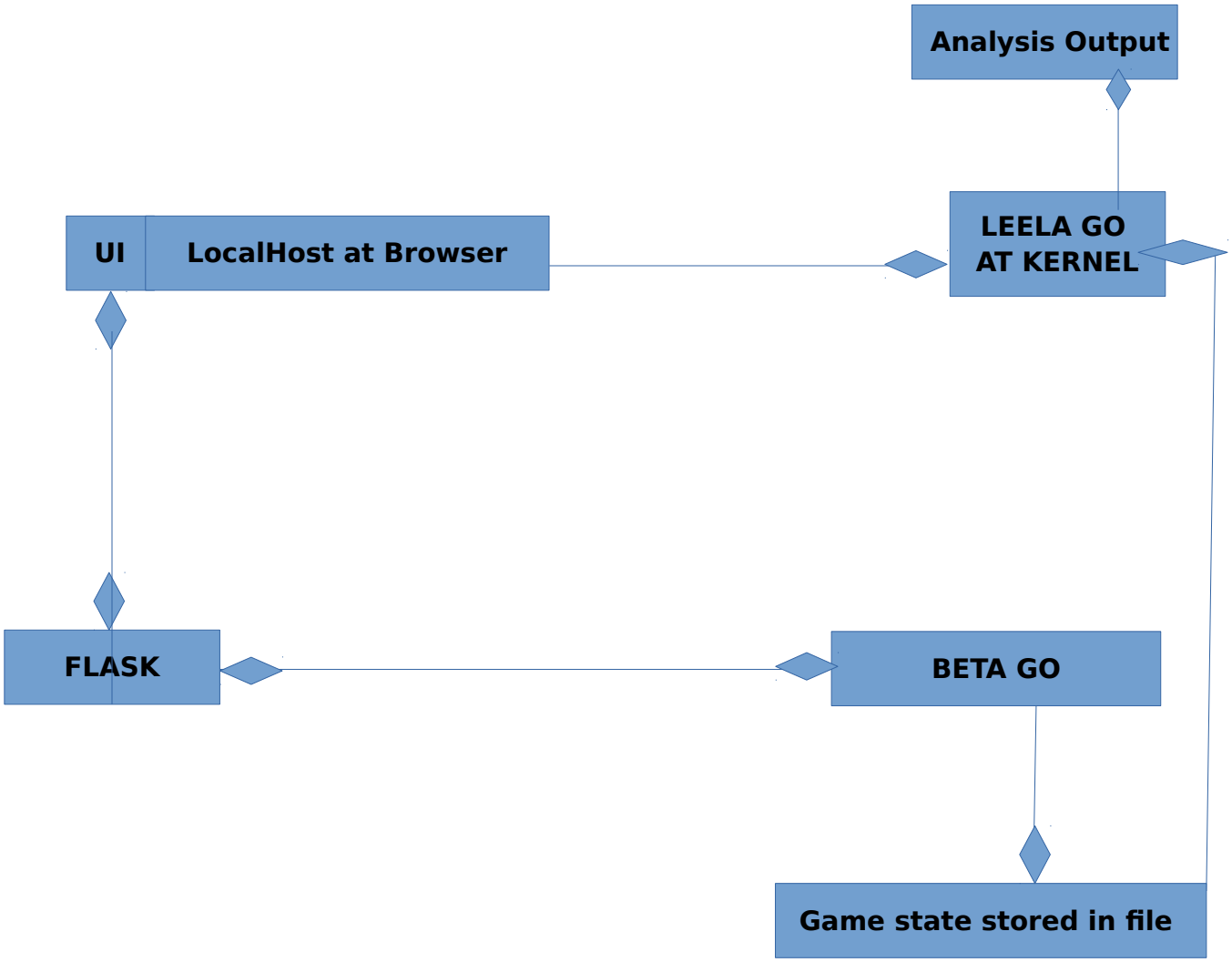
-> BetaGO

-> Leela Game Engine

-> Rochester AlphaGo

WE TOOK 'BetaGo' AND 'Leela Game Engine' into consideration and have joined the plus points of the implementation and have come out with our newer version named as 'MinAlphaGo'

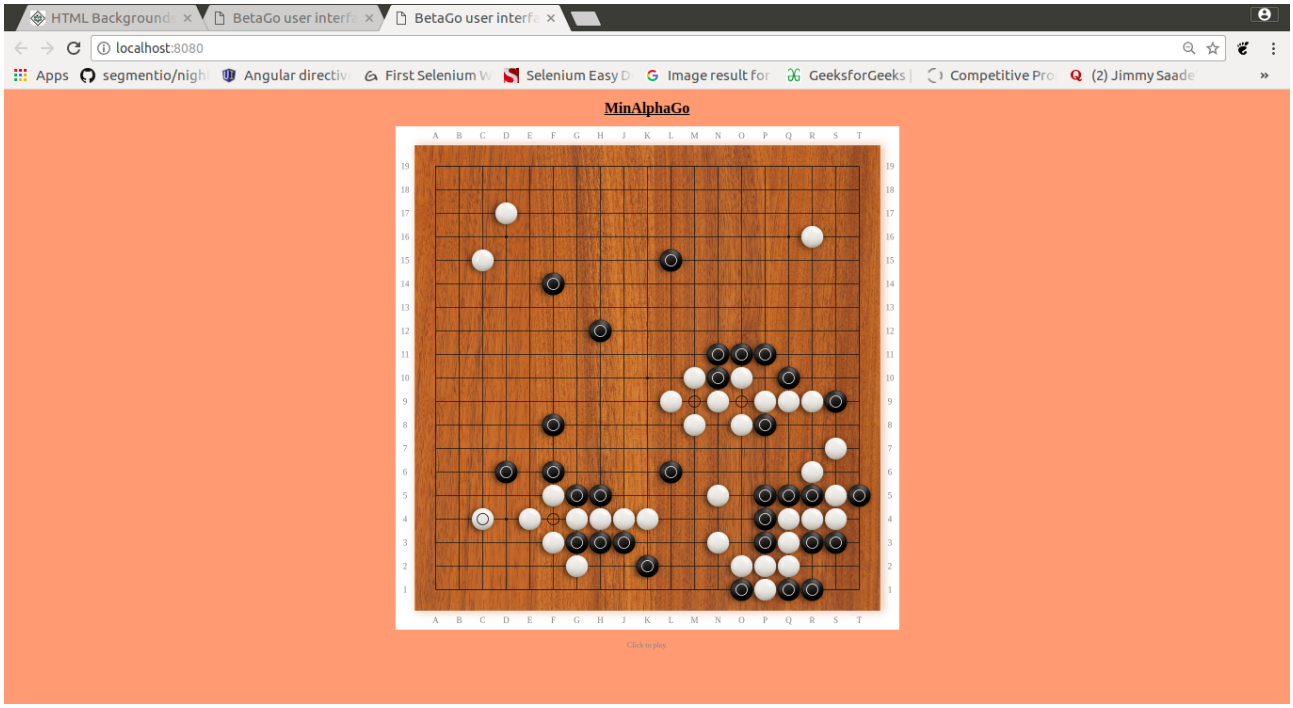
Flowchart of Implementation:



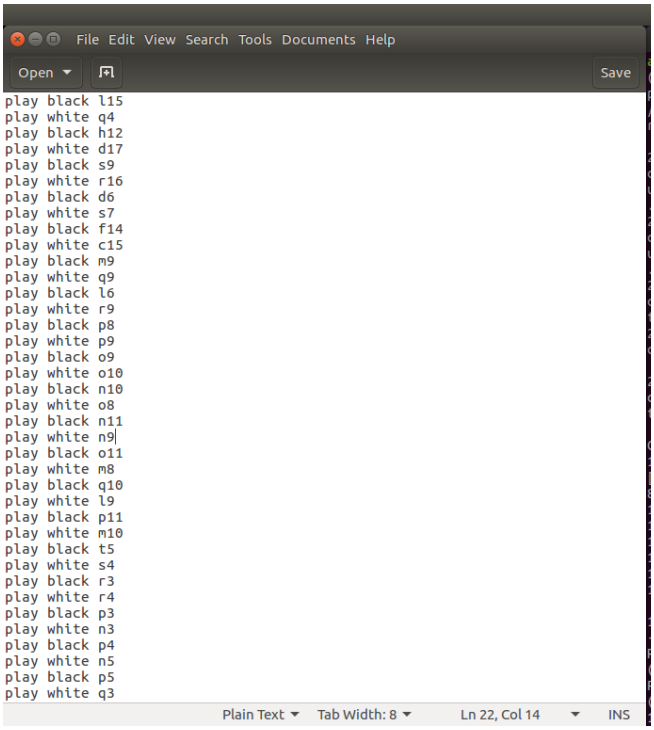
Analysis and Results :

For instance we took a game state :

This is the MinAlphaGo interface running on localhost.

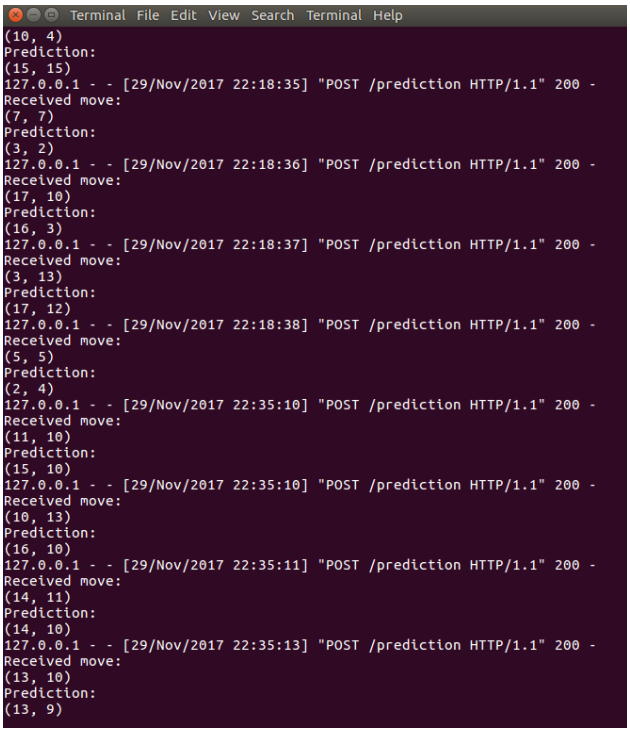


Following image is showing the series of moves that are getting stored in a file ‘moves.txt’ at the backend.

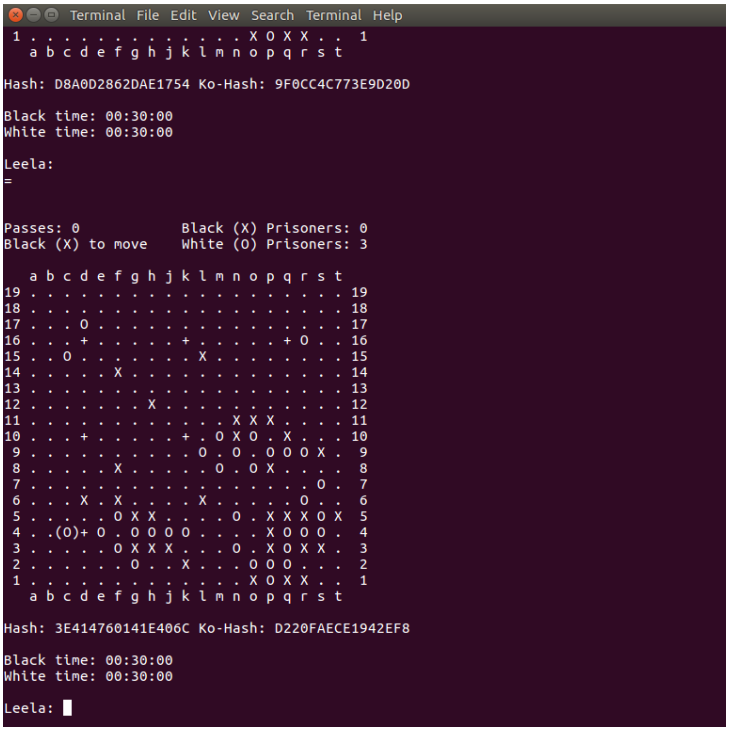


Starting the Server :

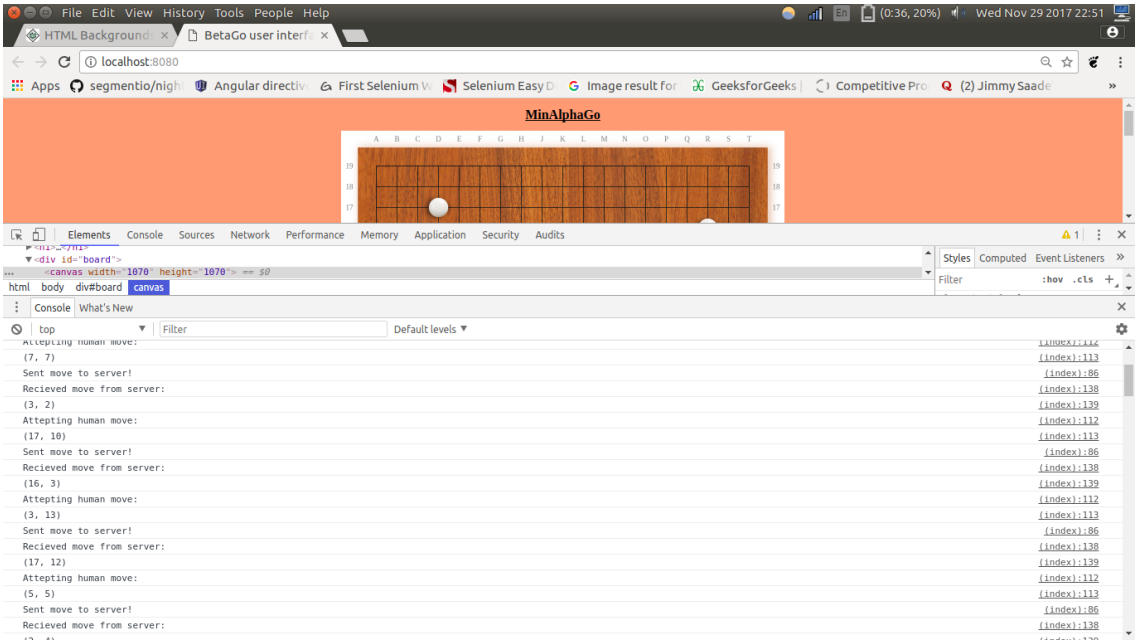
Status of Kernel while we are playing using UI:-



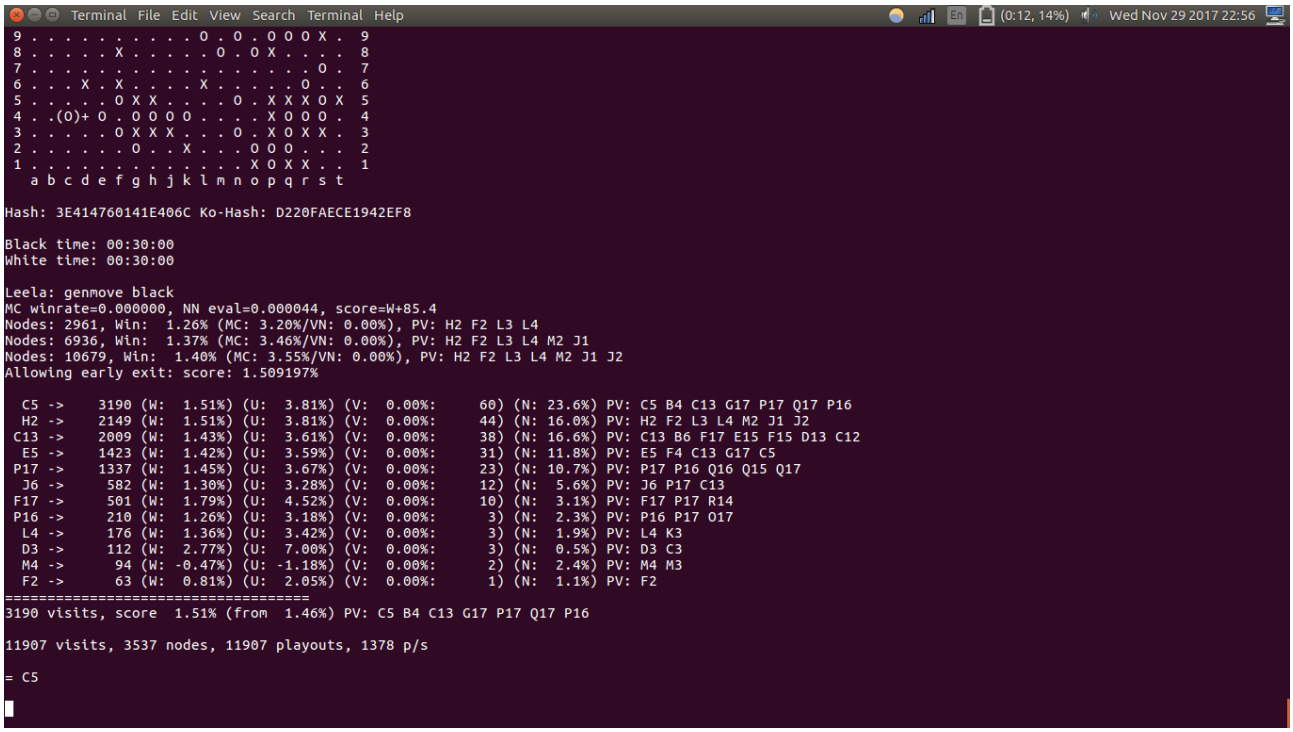
Starting the Leela’s Kernel Implementation...



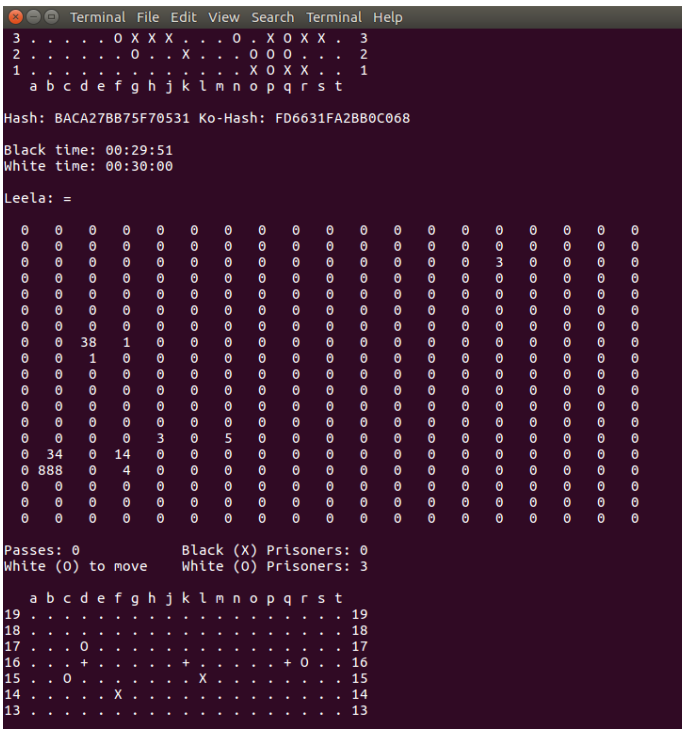
Status of Browser’s Console :



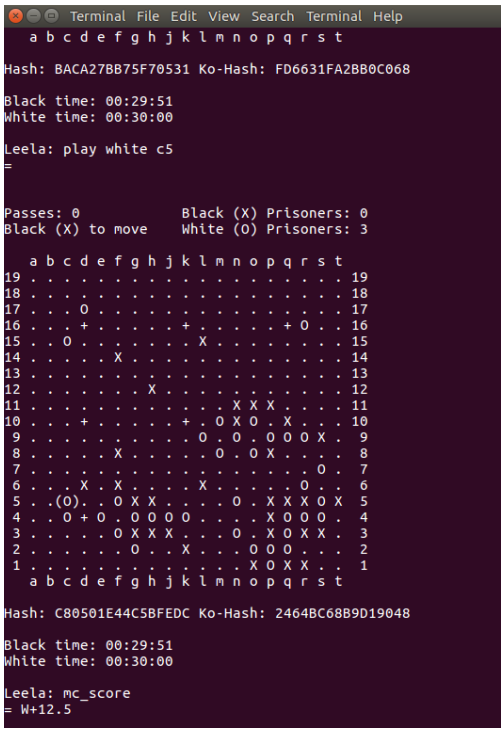
Analysis and best moves prediction :



Heatmap :-



Calculating Win Rate :



RESULTS:

BetaGo:

- UI wasn’t implemented properly.
- No Analysis is done.
- Faster than Leela.
- Only Supervised learning was Used to train the Network.

Leela:

- Has a good User Interface.
- It is comparatively slower than MinAlphaGo.

MinAlphaGO:

- We were able to perform the analysis part successfully.
- It predicts move faster than leela.
- Improved User Interface.

FUTURE SCOPE:

- MCST can also be used to refine the training process.
- Reinforcement learning along with policy and value networks can be used to train the model more efficiently.
- Different bots can be implemented.

List of References:

1. Fortune Article : http://fortune.com/2017/05/23/google-ai-alphago-china-master-ancient-game-go/
2. Machine Learning Videos by Professor Andrew Ng for Machine Learning (CS 229) in the Stanford Computer Science department. : https://www.youtube.com/watch?v=UzxYlbK2c7E
3. Wikipedia :
    a. https://en.wikipedia.org/wiki/AlphaGo
    https://en.wikipedia.org/wiki/Go_(game)
4. 'Google DeepMind's AlphaGo: How it works?' Article by Tastehit https://www.tastehit.com/blog/google-deepmind-alphago-how-it-works/
5. Leela (Game Engine) - https://senseis.xmp.net/?Leela
6. Flask(Web Microframework) - https://www.tutorialspoint.com/flask/flask_application.htm