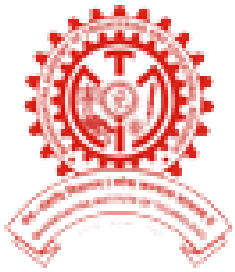
**A SEMINAR REPORT ON**

# AI and Games

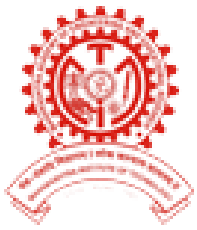
SUBMITTED BY

**Kunal Kolhe**
**Roll No : 3152061**
**Email ID : knl.kolhe@gmail.com**

UNDER THE GUIDANCE OF
**Prof. Pratima Joshi**



**Department Of Computer Engineering**
**MAEERs**
**MAEERs MIT College of Engineering**
**Kothrud, Pune 411 038**
**2017-2018**



**Department of Computer Engineering**

**MIT College of Engineering**
**PUNE**

# C E R T I F I C A T E

This is to certify that Kunal Kolhe from Third Year Computer Engineering has successfully completed his seminar work titled 'AIs for Games' at MIT College of Engineering, Pune in the partial fulfilment of the Bachelors Degree in Engineering.

| | | |
|---|---|---|
| Prof. Pratima Joshi<br>(Seminar Guide) | Prof. Bharti Dixit<br>(Head of Department) | Prof. Anil Hiwale<br>(Principal) |

Place: Pune Date:

## ACKNOWLEDGEMENT

I take this opportunity to express my sincere appreciation for the cooperation given by Prof. Dr. B. A. Dixit, HOD (Department of Computer Engineering) and she needs a special mention for all the motivation and support.

I would also like to thank my guide Prof Pratima Joshi and Seminar Co-ordinator Prof. Bharati Ainapure for their help and guidance in the seminar process.

For all efforts behind the Seminar report, I would also like to express my sincere appreciation to staff of department of Computer Engineering, Maharashtra Institute of Technology Pune, for their extended help and suggestions at every stage.

Kunal Kolhe

(Exam Seat No:3152061)

# Contents

# List of Figures

# List of Tables

**Abstract**

Artificial Intelligence has been used in gaming since the beginning of the development of Artificial Intelligence. AI is used in games because we can easily test out AI algorithms in a small use case similar to the real world. Games represent problems which can be found in real life. We started building AI for small games and as the technology evolved, we are building more complex AI which can not only play very hard games better than humans but also they can play different games without need for adjustment.
We have come a long way since IBM's Deep Blue defeated a reigning world champion in the game of chess. Go is board game thought to be the most complex board game in the universe with more combinations of moves than there are atoms in the universe. It was thought that for an AI to defeat a human at Go was at least a decade away. Google's DeepMind created an AI which using an image as an input astounded everyone by defeating a world champion in the game of Go. This report will cover in brief the working of the AI which defeated 18 times world champion Lee Sedol at the game of Go.

**Keywords:**
Atrificial Intelligence, Google DeepMind, AlphaGo.

# Chapter 1

# Artificial Intelligence for Games

## 1.1 Why do we implement Artificial Intelligence for playing games

We have been developing Artificial Intelligences for games since decades. Instead of building robots and testing AIs for them in the real world, games offer a simulated environment to test small AI systems. These small AI systems can be combined together to operate in the real world. It also allows for testing out new AI algorithms in a cost effective manner with low risk. In Video Games we take Complex problems and reduce them to smaller and more manageable problems.

For example consider navigating a city while walking without bumping into pedestrians, In a simulated environment, we can easily develop an algorithm for the character to roam the city without bumping into obstacles and other people. Making a robot, then testing it in a real world environment would be very hard for this application because first we would have to teach the robot how to walk, then teach it to detect obstacles, then teach collision avoidance strategies. There will be a huge overhead in building the robot and teaching it how to walk. In a video game we just develop a relevant algorithm to navigate an urban setting. Later on this algorithm can be used in a real world robot when the robot has been made.

There is a huge variety of games. If an AI using one algorithm can play one game

and apply the principles it learned from that game to another game, this is a sign of generalized intelligence. The more games the algorithm can be applied to the smarter this algorithm is.

## 1.2 What is the use of General Artificial Intelligence?

The ultimate objective is to build an AI which is truly general Purpose, meaning that it acts similar to a human in terms of training. We still do not understand how human intelligence works. Even though we know the structure of the neurons, the physical map of the brain, we do not yet understand how our brain processes data. Using Deep Learning we are trying to mimic human intuition and trying to solve problems in a heuristic manner. Building a general Purpose AI needs us to formalize intelligence or in other words find out how it works. If we know the formula for intelligence we can also implement it on humans. If we understand how the human brain works we can develop systems to interface with brains; not just read our brains but also to write to them.

Humans cannot handle huge amounts of data. That is why we develop General Purpose AIs to manage data as our world is becoming increasingly interconnected. There can be General Purpose AI for driving all the cars interfacing with another AI responsible for monitoring the city roads, scheduling maintenance, etc. Large systems like cities can be managed effectively by AIs.

On a bit of a side note, Currently human brain is the most efficient processor in the world. The maximum words a human can read is currently between 1000 and 2000 words per minute with 50% comprehension or above. The average word length in English is 5.1 letters per word. A letter is 8 bits and Currently Solid State Drives

(SSDs) have read/write speeds up to 560MB/s then an SSD can read upto 13 725 490.2 words per second. Imagine a human being able to consume that much data. If we truly learn how our brain works then we can just remove all the bottlenecks currently present in the system due to human slowness. What could happen is AI replace human jobs and then again if humans become as cost effective as machines, we could again replace AIs with humans. But that is a discussion for the future.

# Chapter 2

# Understanding the game of Go

Go is an abstract strategy board game for two players, in which the aim is to surround more territory than the opponent.

The game was invented in ancient China more than 2,500 years ago and is believed to be the oldest board game continuously played today. It was considered one of the four essential arts of the cultured aristocratic Chinese scholars in antiquity. The earliest written reference to the game is generally recognized as the historical annal Zuo Zhuan (c. 4th century BC).

Despite its relatively simple rules, Go is very complex. Compared to chess, Go has both a larger board with more scope for play and longer games, and, on average, many more alternatives to consider per move. The lower bound on the number of legal board positions in Go has been estimated to be

$$2 * 10^{170}.$$

The playing pieces are called "stones". One player uses the white stones and the other, black. The players take turns placing the stones on the vacant intersections ("points") of a board with a 1919 grid of lines. Beginners often play on smaller 99 and 1313 boards, and archaeological evidence shows that the game was played in earlier

centuries on a board with a 1717 grid. However, boards with a 1919 grid had become standard by the time the game had reached Korea in the 5th century CE and later Japan in the 7th century CE.

Once placed on the board, stones may not be moved, but stones are removed from the board when "captured". Capture happens when a stone or group of stones is surrounded by opposing stones on all orthogonally-adjacent points. The game proceeds until neither player wishes to make another move; the game has no set ending conditions beyond this. When a game concludes, the territory is counted along with captured stones and komi(points added to the score of the player with the white stones as compensation for playing second, which is normally either 6.5 or 7.5 depending on the rule-set being used) to determine the winner. Games may also be terminated by resignation.

As of mid-2008, there were well over 40 million Go players worldwide, the majority of them living in East Asia. As of December 2015, the International Go Federation has a total of 75 member countries and four Association Membership organizations in multiple countries.

# Chapter 3

# Why was Go considered the hardest game to develop an AI for?

IBM built Deep Blue to play the game of chess. On 10 February 1996 Deep Blue defeated world champion Gary Kasparov. This was the first time a human world champion had been defeated by an AI under tournament conditions. However, Kasparov still won the 1996 series against Deep Blue by a score of 4-2. Deep Blue was heavily upgraded and again a series was played in 1997 in which Deep Blue won by 3 - 2 and becoming the first computer system to defeat a reigning world champion in a match under standard chess tournament time controls. Deep Blue calculated all possible moves resulting from a move m and then chose the best possible move. Deep Blue was based on brute force and it won because technology had evolved enough that Deep Blue could compute faster than a human. Deep Blue could explore upto 200 million chess positions per second. Some games may be solved by recursively computing the optimal value function in a search tree containing bd possible sequences of moves, where b is the games breadth (number of legal moves per position) and d is its depth (game length). For chess b is approximately 35 and d is approximately 80. For Go b is approximately 250 and d is approximately 150. Which is a huge number. There are more possible combinations for the game of Go than there are atoms in the universe. It was considered that beating a human champion at Go would take at

least 10 years. AlphaGo surprised the entire AI community.

# Chapter 4

# Understanding how AlphaGo works

As mentioned earlier, depth search of all possible moves is impossible. But, the effective search space can be reduced by 2 general principles.

First, the depth of the search tree may be evaluated by truncating the search tree at state s and replacing the subtree below s by an approximate value function for that entire subtree.

Second, the breadth of the search may be reduced by sampling actions from a policy p(a—s) that is a probability distribution over possible moves a in position s. For example, Monte Carlo rollouts search to maximum depth without branching at all, by sampling long sequences of actions for both players from a policy p. Averaging over such rollouts can provide an effective position evaluation, achieving weak amateur level play in Go.

The focus of Monte Carlo tree search is on the analysis of the most promising moves, expanding the search tree based on random sampling of the search space. Monte Carlo tree search (MCTS) uses Monte Carlo rollouts to estimate the value of each state in a search tree. As more simulations are executed, the search tree grows larger and the relevant values become more accurate. The policy used to select actions during search is also improved over time, by selecting children with higher values. As number of simulations becomes very large, this policy converges to optimal play, and

the evaluations converge to the optimal value function.[2]

## 4.1 Monte Carlo Tree Search Algorithm

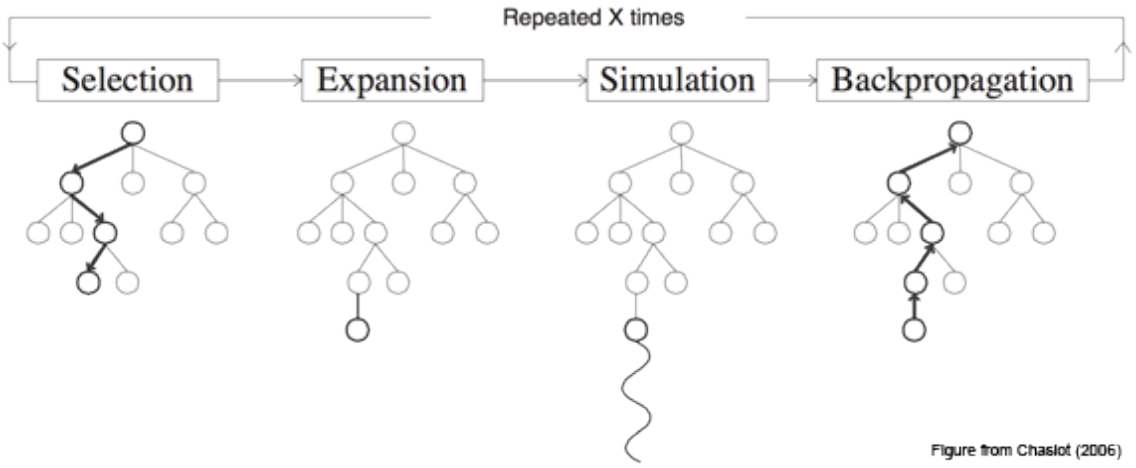



**Figure 4.1:** Monte Carlo Tree Search Algorithm

The Monte Carlo Tree Search Algorithm consists of 4 stages which are repeated indefinitely.[6]

- Selection
- Expansion
- Simulation
- Back propagation

We use Monte Carlo Tree Search Algorithm in most games because its time complexity is not dependant on the total stages available for play.
MCTS chooses branches at random and that becomes an important point why this algorithm is used as a search algorithm for most games having a state tree.

### 4.1.1 Selection

In Selection Stage the paths of the tree are chosen using a policy network.Selection is done till the policy gives a definite result.

### 4.1.2 Expansion

Once the network reaches a part of the tree from where there has been no previous expansion, it chooses paths at random.

### 4.1.3 Simulation

Once a path has been chosen at random, as many paths as possible are played out within the time limit. All the probabilities on those paths are calculated.

### 4.1.4 Back propagation

All these probabilities found in later stages of the tree help to determine overall probability of the leaf nodes and doing this recursively we get an overall probability for moves quite close to the leaf node.

## 4.2 Machine Learning stages

AlphaGo is given an input state s of the board at that time as an image. They (Deep Mind) use neural networks to reduce the effective depth and breadth of the search tree: evaluating positions using a value network and sampling actions using a policy network.

They train the neural network using a pipeline consisting of several stages of machine learning.

### 4.2.1 Pipeline Stage 1:

They begin by training a supervised learning (SL) policy network directly from expert human moves. This provides fast, efficient learning updates with immediate feedback and high-quality gradients.

### 4.2.2 Pipeline Stage 2:

Next, they train a reinforcement learning (RL) policy network that improves the SL policy network by optimizing the final outcome of games of self-play. This adjusts the policy towards the correct goal of winning games, rather than maximizing predictive accuracy.

### 4.2.3 Pipeline Stage 3:

Finally, we train a value network that predicts the winner of games played by the RL policy network against itself. The program AlphaGo efficiently combines the policy and value networks with MCTS.

## 4.3 Supervised Learning of Policy Networks

DeepMind Trained a 13 layer policy network from 30 million Go games from the KGS Go server. The Network predicted expert human moves correctly 57% of the time. Compared to the other State-of-the-art Go AI at the time which could predict human moves only 44% of the time, this SL policy network was more accurate and took only 3ms to predict a move.

## 4.4 Reinforcement Learning of Policy Networks

This network is the same as the SL network initially. Each move is assigned a value v before the network is trained. We play games between the current policy network and a randomly selected previous iteration of the policy network. Randomizing from a pool of opponents in this way stabilizes training by preventing overfitting to the current policy. A reward function rewards moves according to whether the Game is won or lost. When a move is chosen, the game is played out to the end so that the network can assess which moves are more likely to result in a win. If a move results in a win, then the next time it will be more likely that the move will be chosen.

## 4.5 Reinforcement Learning of Value Networks

The final stage in the training pipeline is the training a value network. The value network gives a probability of which player is more likely to win given the state of the board at the current position.

Ideally the result of the value network would be based on optimum play, but in practice the result of the value network is based on the currently strongest policy. This neural network has a similar architecture to the policy network but outputs a single probability instead of a probability distribution.

## 4.6 Searching with policy and value networks

AphaGo Uses a Monte Carlo Tree Search Algorithm to read sequences of future moves effectively. The Monte Carlo Tree Search Algorithm chooses a branch randomly of the tree of all possible future moves. This is to avoid finding a good move after a long

time of searching the tree because the search was conducted in an ordered manner. AlphaGo combines the policy and value networks in an MCTS algorithm that selects actions by lookahead search. Each edge of the search tree stores an action value, visit count, and prior probability. The tree is traversed by simulation(that is, descending the tree in complete games without backup), starting from the root state. At each time step of each simulation, an action is selected from state so as to maximize action value plus a bonus that is proportional to the prior probability but decays with repeated visits to encourage exploration.

## 4.7 To sum it up

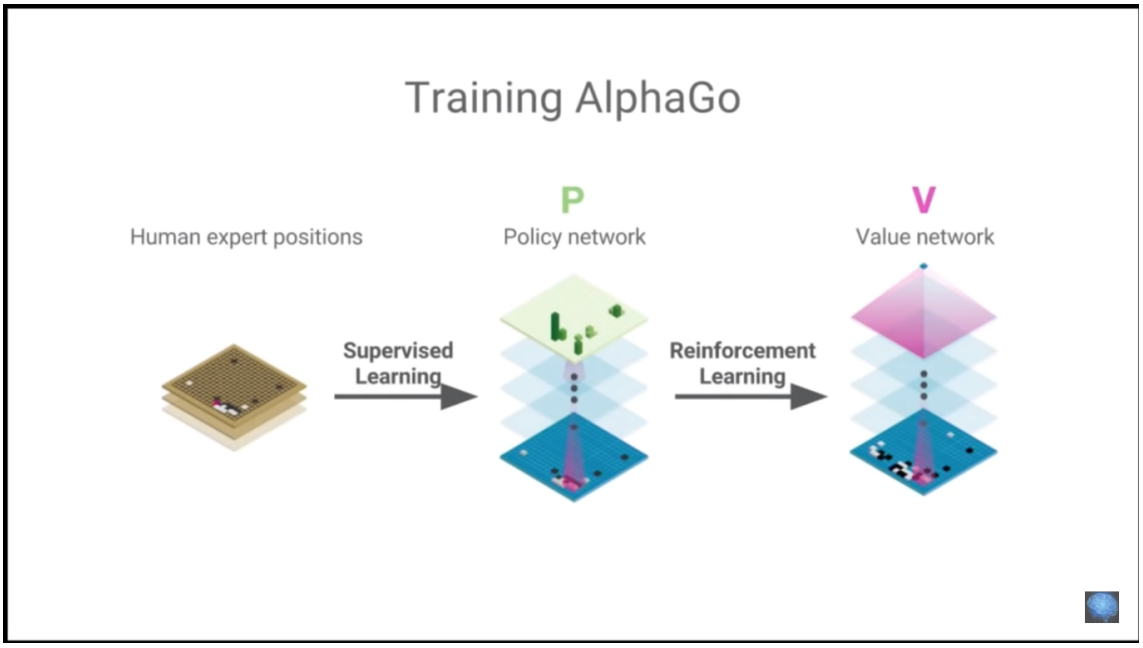



**Figure 4.2:** AlphaGo Training visualized

Therefore, AlphaGo plays a game using these stages:

First, after being trained it evaluates the board and chooses a few specific moves to play.

For each move chosen, it either evaluates the quality of the move by using a value

network after that move or running a deeper Monte Carlo lookahead search, using the policy network to speed up the search.

It evaluates the state of the board after choosing each move and determines if the probability of winning is in its favour or not.

## 4.8 Hardware used by AlphaGo

The final version of AlphaGo used 40 search threads, 48 CPUs and 8 GPUs.

There was another distributed version of AlphaGo that exploited multiple machines, 40 search threads, 1202 CPUs and 176 GPUs.

# Chapter 5

# AlphaGo Zero

AlphaGo Zero still uses a Monte Carlo Tree Search algorithm to perform searches, but instead of a separate Policy network to choose a move and a value network to evaluate the moves, AlphaGo Zero combines these 2 components. AlphaGo zero is not trained on previous human matches. Zero instead learns to play by playing games against itself starting from completely random play. The neural network starts off knowing nothing about the game of Go. AlphaGo Zero combines its own random play with search algorithms and begins to learn from itself. [5] After only 3 days of training, AlphaGo Zero was able to beat AlphaGo Lee which defeated legendary Go player Lee Sedol.

# Chapter 6

# Lee vs Alphago

The creators of AlphaGo arranged a tournament standard series of matches between AlphaGo and the legendary Go player Lee Sedol. Lee Sedol has won 18 international titles as of February 2016. In this chapter we discuss the matches between Lee and AlphaGo [3]

## 6.1 Game 1

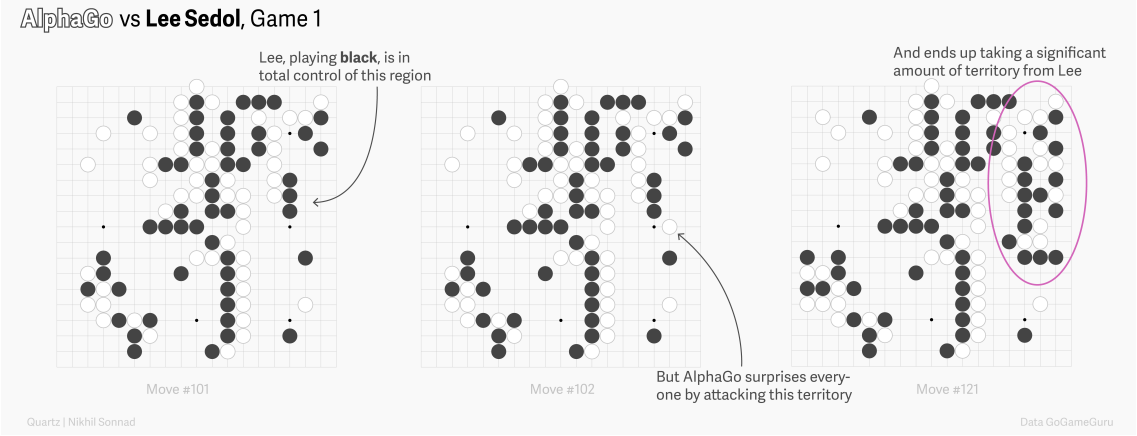



**Figure 6.1:** game 1 of AlphaGo vs Lee Sedol

Lee was doing well, and the two were engaged in a skirmish on the left side of the board. But AlphaGo, playing with the white stones, suddenly attacked deep inside Lee-controlled territory on the right side.

This was totally in the black area, Leesaid. Human players would never think about doing that.

Lee responded, quickly capturing three of AlphaGos stones. It was a poor move by AlphaGo, or so it seemed.

Twenty moves later, AlphaGo had taken three of Lees stones in the upper right and occupied about half the area that most human observers had written off as impregnable. Sacrificing three stones turned out to be a key pivot, turning the game in AlphaGos favor.

Even in blacks area, white got a result. Its unacceptable for black, Leesaid. There are huge variations in a Go game, we cant even read 1% of them. We have certain patterns in our minds when we play, so this is the kind of move we would never think about.

## 6.2 Game 2

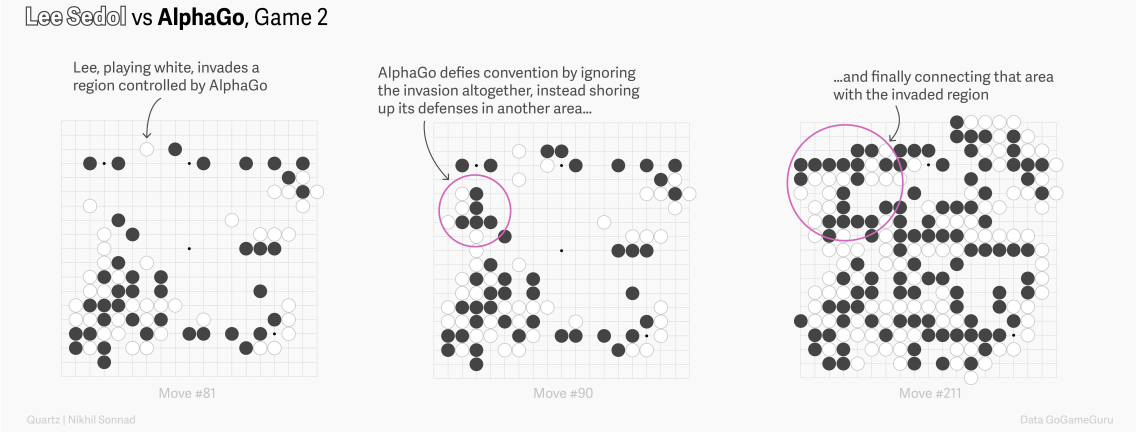



**Figure 6.2:** game 2 of AlphaGo vs Lee Sedol

Lee began an invasion of AlphaGos territory, but the AIs response was unexpected: It seemed simply to ignore Lees attack. That turned out to be a smart move.

Its like your opponent broke into your house and wants a big fight with you, but you go and make a cup of coffee first, Lee said. Instead of immediately defending itself, AlphaGo first strengthened its defenses to ensure that Lee wouldnt gain much territory in the attack.

In the end, AlphaGo secured two areasat the top-left of the board and just below itsuccessfully limiting Lees incursion. Blacks territory is still blacks territory, although white has a weak group there. From here, the game is already finished for Lee Sedol. Theres no chance after this, Leesaid. Lee Sedol played in a normal way, but AlphaGo answered in an unusual way.

## 6.3 Game 4

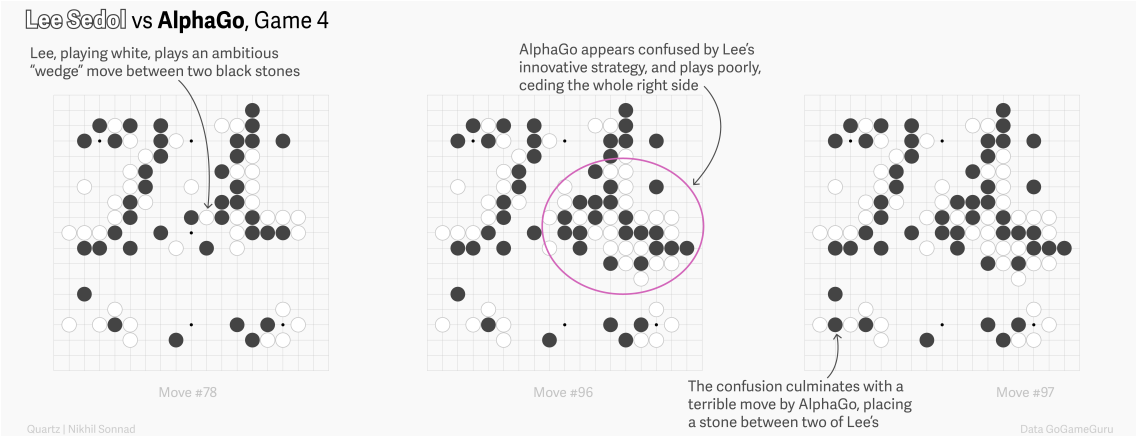



**Figure 6.3:** game 4 of AlphaGo vs Lee Sedol

In their fourth game, the only one in which Lee was victorious, he appeared to adopt some of AlphaGos strategy by pursuing less expected and riskier maneuvers that proved successful in the end.

Lee played a wedge move, placing his white stone between two of AlphaGos. This

is generally avoided since the point of Go is to surround the other players stones, and a wedge move is essentially the opposite. But Lee did so right in the middle of the board, puzzling observers.

Its hard to say if it was a correct move or not, Lee said.

AlphaGo couldnt interpret it, either. Thrown off by the wedge move, the AI made a series of amateurish mistakes. Lee Sedol found a move that was out of AlphaGos thinking, Leesaid.

Seven moves after Lees wedge, AlphaGo had lost its grip on the right side of the board. The AI attempted a wedge move of its own, but it didnt make any sense in the context of the game. Its like an amateur players level, Lee said.

Lee went on to win the fourth game. AlphaGo regained its composure to win the fifth and take the match, 41. But that brief moment of unusual and effective strategizing by Lee demonstrated that the true value of artificial intelligence reaches far beyond the simplistic narrative of man versus machine. Instead, AIs potential may be in teaching humans new ways of thinking for ourselves.

## 6.4 Lee after the series

After his fourth-match victory, Lee was overjoyed: "I don't think I've ever felt so good after winning just one match. I remember when I said I will win all or lose just one game in the beginning. If this had really happened I won 3 rounds and lost this round it would have had a great bearing on my reputation. However, since I

won after losing 3 games in a row, I am so happy. I will never exchange this win for anything in the world." He added: "I, Lee Se-dol, lost, but mankind did not." After the last match, however, Lee was saddened: "I failed. I feel sorry that the match is over and it ended like this. I wanted it to end well." He also confessed that "As a professional Go player, I never want to play this kind of match again. I endured the match because I accepted it."

# Chapter 7

# Other notable AIs developed for simulated environments

## 7.1 OpenAI bot for DOTA 2

In 2017s yearly tournament of DOTA 2(Defence Of The Ancients is a video game), OpenAI, a start-up backed by Elon Musk presented a bot that would defeat Human Pro players in a 1-vs-1 tournament. OpenAI says that the bot learned enough to defeat Pro players after just 2 weeks of training.

Elon Musk tweeted : OpenAI first ever to defeat world's best players in competitive eSports. Vastly more complex than traditional board games like chess & Go

## 7.2 Google DeepMind walking bot

Googles DeepMind has developed an AI that taught itself how to walk.

They set a point A in a simulated environment and set another point B in that environment. The only input to the bot was to go from point A to point B. The bot was given a few virtual sensors (for example whether it is upright or not) and then it was incentivised to move forward.

Using Reinforcement Learning, the bot taught itself how to move from point A to B while walking. Then the environment was made harder with obstacles to go around

and holes to go over. The bot even mastered these environments through trial and error.

# Chapter 8

# Where do we go from here?

As discussed earlier, in the match of Lee Sedol vs AlphaGo Sedol played a very uniquely human move(Creative) which threw off AlphaGo. This resulted in Sedol winning the match. AIs can amass lifetimes of experience in just a matter of days. One way we can benefit from AIs is to learn from them. Even OpenAIs bot is being used to learn maneuvers which we never thought to try. The same goes for chess AIs. Players play hundreds of matches against Chess AIs to improve their games. Using Reinforcement Learning, more generalized AIs are being made which can perform different tasks by training them for that task.

Behaviour tests conducted on Google's DeepMind AI system make it clear just how careful we need to be when building the robots of the future.[8]

Then it started figuring out how to seamlessly mimic a human voice.

More recently in 2017, researchers tested its willingness to cooperate with others, and revealed that when DeepMind feels like it's about to lose, it opts for "highly aggressive" strategies to ensure that it comes out on top.

Commercial Applications:

- The new Google Cloud Text-to-Speech application programming interface costs $16 for every million characters of text it processes in DeepMind's artificial male and female voices.

- In 2016, Google said it had drastically decreased the expense of cooling its data centers using DeepMind's wizardry.[9]

DeepMind : As well as the immediate potential to help clinicians provide better, faster and safer care, were excited about the potential to one day help with some of the other challenges facing healthcare systems, including empowering patients to look after themselves and their families health, and supporting coordinated ongoing care around patients needs.

# Chapter 9

# Conclusion

This is a very exciting time to observe the developments made in Artificial Intelligence. Most limitations in the development of AI are related to the Algorithms and software. Many organizations are coming together to form pacts to ensure Artificial Intelligence is not used to harm humans. OpenAI is one such initiative whose objective is to make AI open source so that big corporations are not the only ones who have access to and control of such technology.

# Bibliography


[1] https://deepmind.com/applied/

[2] https://www.nature.com/articles/nature16961

[3] https://qz.com/639952/googles-ai-won-the-game-go-by-defying-millennia-of-basic-

[4] https://www.britgo.org/intro/intro2.html

[5] https://deepmind.com/blog/alphago-zero-learning-scratch/

[6] https://en.wikipedia.org/wiki/Monte_Carlo_tree_search

[7] https://deepmind.com/applied/deepmind-health/

[8] https://www.sciencealert.com/google-deep-mind-has-learned-to-become-highly-aggr

[9] https://www.cnbc.com/2018/03/31/how-google-makes-money-from-alphabets-deepmind-
html