Technical report

CO600: Project

School of Computing

Supervised by Radu Grigore

<http://machinelearningxo.github.io/>

*Basiru Andrew, Paul Crawford, Met Dekaj,*

*James Hannigan and Tahmoor Ulhaq,*

# 1.1. Abstract

The aim of our project is to create an

Artificial Intelligence which plays Tic Tac Toe and Gomoku board game against either a computer or player opponent. We intend to do so in a fully interactive website available online - whereby users can play Tic Tac Toe or Gomoku on the same site.

The Artificial Intelligence architecture of the game will be a variation of the Minimax algorithm which predicts the next move and chooses the best possible option on the board. This will work for Tac-Tac-Toe but due to the fact that Gomoku has many potentials moves we are also implementing alphabeta pruning, as this will make the algorithm more efficient. Doing so by avoiding unnecessary paths when calculating moves in the tree search.

# 1.2. Introduction

The Artificial Intelligence(AI) field has been growing and attracting more and more attention among different industries and the general population for its problem-solving capabilities, with examples like automated robot arms used in assembly lines, or simply personal assistants like Cortana (by Microsoft) of Siri (by Apple) which are increasingly been used for everyday life tasks.

We have taken a particular interest and inspiration for this project from Google's AlphaGo AI which implements a system that learns how to play using the MonteCarlo algorithm, that uses randomness to calculate the best next move. But we decided to use another AI algorithm called Minimax created by the mathematician/Computer Scientist,

“John Von Neumann” (1903-1957) in 1928 marking the beginning of game theory [12].

First the most important thing for the group was learning and understanding the topics of neural network and machine learning, so we did some research and found out about depth-first search algorithm, and then progressed into researching the Minimax algorithm, which looked like a good algorithm to use for a game like Tic Tac Toe as it added the advantage of backpropagation to the depth-first search (more information in background[1.3.3]). While researching more about the Minimax algorithm we also found out about Alpha Beta Pruning which decreases the number of nodes the minimax algorithm has to visit, making it quicker and more efficient.

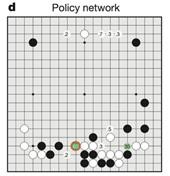
This report will be detailing the development and implementation of Machine Learning XO. Section 1 is focused on introducing the project in an abstract view and detailing the basis of our idea from DeepMind’s implementation of AlphaGo. Within this section we also provide a technical introduction on; neural networks, tree data structures and searching algorithms, Minimax, Monte Carlo, Brute force and Alpha-Beta pruning. Section 2 further describes the implementation process of these algorithms in our project in technical detail and denotes sections on design, features, challenges, consistency and results. Section 3 focuses on the quality assurance of the project --and the processes we took to ensure it. Section 4 focuses on the evaluation of the project, processes and future developments.

1.2.1. DeepMind’s AlphaGo

Our project was based on ideas of

DeepMind’s AlphaGo technology which is the first program to defeat a professional human Go player and a Go world professional by using a mixture of deep learning and reinforcement learning. In the game Go, two players take turns putting their pieces on spots where lines cross on a grid this is usually on a 19 x 19 board, which made it perfect for our Gomoku (5 in a row) game. The Main aim of the Go game is to secure territory by surrounding other pieces and restricting their movement where you then are able to collect the opponent’s piece. This is a very strategic game which ultimately ends when both players decide they are not able to make any more progress. The complexity of the game helped us to understand how our Tic Tac Toe game should run, in Tic Tac Toe 2 players take turns placing their selected symbols (X or O) on a 3 x 3 board, when humans play this game we do not enumerate through every possible board layout but instead use our intuition as a sort of look ahead technique.

AI has different techniques for this, one of them is the MiniMax algorithm which we have used for Tic Tac Toe, another algorithm which AlphaGo uses is the Monte Carlo tree search which finds its moves based on knowledge and will be the most appropriate for our Gomoku game. There have been multiple versions of AlphaGo with each one improving on its predecessor, the first version used two neural networks a policy network which reduces the number of possible moves and a value network which works out the likelihood the position on the board will lead to a win. They work together to enable them to choose their moves and are known as convolutional neural networks (CNN) which are able to take images as inputs and output class probabilities after being trained on a labelled image dataset thus learning the mapping between inputs and outputs.



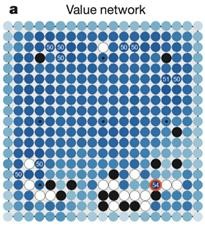
*Figure 1: Policy network diagram [10]*

The policy network takes board positions as inputs and decides on the next best move to make. The numbers indicate the likelihood of a human player placing their next piece on that spot[10].

The DeepMind program trained the policy network on millions of examples of moves made by good Go players this enabled it to match moves that good human Go players would make with 57% accuracy[15]. They improved this with reinforcement learning by simulating the game to the end point, in order to teach the network which moves would lead to winning the game instead of teaching it only the moves a human player may choose to make. Although it was fast enough to pick one good move it still needed to check through thousands of possible moves they again improved on this by changing the network so it did not look at the entire 19 x 19 board, but instead look at a smaller

section around the opponent’s previous move and the next move they are making.

The value network estimates the value of each player winning the game based on the board position, enabling it to judge future board positions as good or bad. When the network evaluates that a position is bad it is able to skip any more moves following that play.



*Figure 2: Value Network diagram [10]*

In the value network, the darker pieces indicate the position for a more likely win for the player.

In addition to the policy and value network, AlphaGo uses an algorithm called the Monte Carlo tree search to help read future moves more effectively. AlphaGo uses these 3 components (policy/value network and Monte Carlo algorithm) altogether once it is trained to play a game:

· From the current board position, it selects possible effective moves using the policy network

· Then evaluates the quality of the move using the value network or the Monte Carlo algorithm with the policy network to speed the search.

The latest version of AlphaGo (AlphaGo Zero) still uses the Monte Carlo algorithm but combines two neural networks to make one neural network which learns to play games by playing against itself thus avoiding having to learn to play through thousands of human games [13].

# 1.3. Background

1.3.1. Neural Network An artificial neural network is layers of nodes all connected to each other and is very similar to the networks of neurons in the brain [1]. The neurons in the neural network will learn to respond to and detect various input patterns that they are presented with. To get this you need to train the neural network with data with known correct solutions so you know how far the output of the neural network was from the actual. These values will then be passed onto an optimisation algorithm which will try to alter the weights between the neurons so better results can be achieved. In a basic neural network, there will normally be layers upon layers of neurons which will take in information from neurons in the previous layer and output to neurons in the next layer. The neurons both have a weight and a bias. The weight values are changed in training and are what causes the network to learn. The bias values are used to adjust how early an activation function will activate [2].

1.3.2. Tree Data Structure and

Searching algorithms A tree is a data structure made up of nodes and edges. A non-empty tree consists of a root node and most likely many other levels of additional nodes. The top node is the parent node and then the nodes below are the children nodes. the children nodes have no clue about the parent nodes. Each node usually will represent some form of data and the edges between these nodes can also carry data, such as weights. This is what is used in neural networks and there are many searching algorithms to get to the correct part of the tree [3].

1.3.3. Minimax

This is a search algorithm used to pick the next move in an n-player game. In a zero-sum game which is a game where each person's gain or loss is exactly balanced by the loses or gains of the other person [4].

|  |
| --- |
| *if(depth == 0 OR gameOver() == true)*  *{*  *init move = previousMove;*  *previousMove.setValue(evaluate(board));*  *return previousMove;*  *}*    *init returnMove; init bestMove;*  *while(moveList.hasNext())*  *{*  *init currentMove = moveList.next(); placeMove(currentMove, player); returnMove = miniMax(depth - 1, nextPlayer); undoLastMove();*  *if(bestMove == null OR bestMove.value < returnMove.value)*  *{*  *bestMove = returnMove;*  *bestMove = currentMove;*  *}*  *}*  *return bestMove;* |

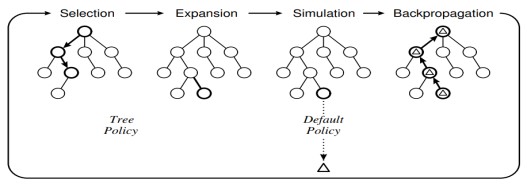
*Figure 3: Pseudo of how we intended to use Minimax*

This works by firstly checking if the depth is 0, so if it's searched far enough to what you wanted, or if the game is over, no more positions on the board or there is a winner, it'll then return the move with a value. The value is used to see if it's the best move to make. Combining this with another heuristic search algorithm should allow us to produce a learning Intelligent system.

1.3.4. Monte Carlo Tree Search

This is a heuristic, problem-solving technique used when classical methods are too slow, search algorithm used mainly in decision processing [5]. There are four main steps to the Monte Carlo tree search. Selection, simply go down the tree picking the best child nodes to do the decision processing the best way. Furthermore, if the decision does not come to an end (win/loss), then create more nodes to try and get to the ending condition. Simulation, this actually plays out the rest of the decision it is processing and finds a way to the end. Back-Propagation, this is when it goes back up the tree updating the tree as it has now found the end [6].

The more simulations are run the more accurate the algorithm becomes.



*Figure 4: Monte Carlo tree search diagram [9]*

Above is a visual representation of the process the Monte Carlo algorithm goes through.

*“Each node is a state, each iteration adds a node, and each edge is an action leading to the next state” [9].*

1.3.5. Brute Force This is a problem-solving technique in which you check every possible scenario and checking if each one will get you to the end result [7]. Therefore, you'd need to input all the end results so that it can check against them.

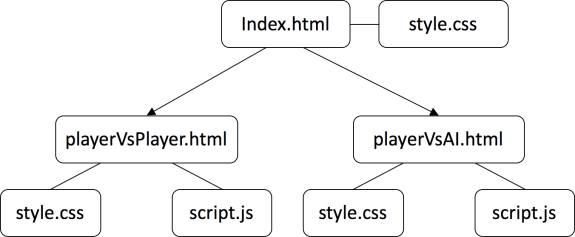
1.3.6. Alpha-Beta Pruning This is an optimisation algorithm used to reduce the number of nodes that are visited when using the Minimax algorithm. It will stop searching when it proves that a move is already worse than any other previous move that you've already found [8].

# 1.4. Aims

Our project aim was to create an artificial intelligence capable of learning to play the game of Tic Tac Toe and subsequently be able to play the best move possible, we also had the ambition of having this system available on an online webpage, so it could be easily accessed by different users. We also wanted to develop the possibility of having a Player vs Player mode for both the games of Tic Tac Toe and Gomoku, which has a bigger board(19x19) and a winning combination of 5 in a row.

# Development

## Design



*Figure 5: Organisation of our files*

We planned originally to code the functionality of the game (solely the

board) and then add a web page’s GUI interface around that board. However, once completing the functionality of the player vs player game modes, we realised they would not be compatible with an AI, without a lot of reconstruction of the code.

Therefore, we decided to keep the player vs player code and begin work on a player vs AI from the beginning. To incorporate both of these games on one website without the confusion of methods and variables with the same name, etc. we used iframes on the main HTML file to include the game boards. This work very effectively, allowing both games to run simultaneously and keeping the code comprehensive.

## Features

Due to the nature of the competitiveness of gamers. It was an important feature to add a record of wins, draws and loses. This was achieved by incrementing variables in the game each time the game has been completed. Thus, in the player vs player mode, users do not need to remember the score, likewise, in the player vs AI game mode.

To make the design of the web page user-friendly and stylish, we want to avoid having too many games on one webpage, thus creating an unorganised and crowded interface. Therefore, we incorporated the use of buttons which modified our player vs player board to change the number of squares it had (from 3x3 to 19x19).

Figure A.1 (in the Appendix) shows the ending conditions that the computer needs to play to.

Figure A2, Figure A3, Figure A4, shows how we implemented the Minimax algorithm in our Tic Tac Toe game.

On line 55, Figure A.2, we start the algorithm by getting all the possible empty spaces that are left on the board, so we know where we can place another piece. The next part from lines 57 to 71 are the evaluating of the board, so if the player1 would win it’ll give a score of 10, if the computer wins it’ll give a score of 10 and if the game ends in a draw it’ll just return a 0. From this we can check to see the next best move as if it returns a -10 you know that the other player will win.

In Figure A.3, is the code where we give each move a value which will be used to pick the next best move. We iterate through the possible squares list and play a game on a new board which is created to give the move from the list a value. By the end of this, the moves array will be populated with moves, with their value of whether they are going to win or not.

The final part of the search function is shown in figure A.4 which is actual chooses the best move to play, the main bit of the Minimax algorithm. This compares the scores that were just given to the move. It does this by first starting off with a number that is really low, usually negative infinity, and checks to see if the score of the current move is greater. If the score is greater it is then updated with the best score and then iterates through the rest of the moves and find the best move. In the end, it returns the best move to play.

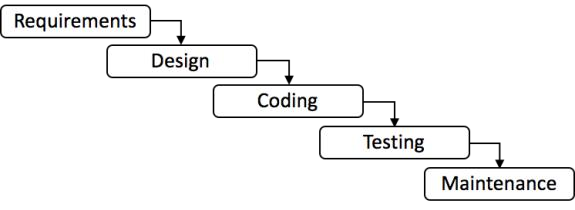
## Challenges

The system originally checked whether a player has won by comparing each state of the board through a set of if statements which would check the horizontally, vertically and diagonally. This was not sufficient for when we developed the Gomoku game as we would need a huge amount of if statements. Therefore, to minimise the number of sequences to check the wins of the game we implemented a process which would iterate through the entire board and check, from a specific square, if there five squares are selected horizontally, vertically or diagonally.

Changing the check winning process to be more efficient and compatible with Gomoku proved to be very complex. This was because the methods needed to be checked in each direction of the win, as well as being able to repeat these methods on each square of the table. We overcome this challenge by programming each part, testing each method as well proceeded and then assembling all the components.

Initially, we did not want to use a brute force approach, so we did not determine predefined winning states for our AI game mode. However, upon programming the project and observing tests of the game; there did not seem to be any process to what moves the AI made; instead it was random. To overcome this, we remodelled the entire file and determined the winning states in the variables of the file. The result of this was that the AI would now select squares to eventually win against the opponent.

## Development Process



*Figure 6: Waterfall Software Development*

During the process of our project, we used the waterfall software development for the development of our project. This works best for us as we made one product, evaluated it and then improved on that. Rather than repeating the cycle of creating the product each time. The most appropriate feature of the waterfall software development method was the

flexibility that it gave us; because of this, we had the opportunity to repeatedly test and make improvements our application.

## Change of programming language

It was important for us to have a stylish GUI interface as we were producing a game. Initially, the game was to be programmed in Java, however, after trying to implement a compatible interface; we decided to develop our project in HTML, CSS and JavaScript. Thus, making the design element less complex and easier to use.

An initial idea before changing to the internet-based languages was to continue working on Java and use a Java applet to display the Java application on a website. This idea proved to be too complicated due to applets now being outdated and unused by most browsers.

By implementing this change, in the final development of the project, we were able to incorporate both the player-vsplayer game and the player-vs-ai by simply adding them in an iframe (as mentioned before). To sum up, these web-based programming languages gave us the freedom to produce a userfriendly interface.

## Consistency

Throughout all of our deliverables, we wanted to keep a consistent design. From the same colour blue being used on our poster as our website. To the

‘Machine Learning XO’ logo being used in this report. This was important for us as a consistent design has proven to evoke an emotional response, increase a system's usability and reduces confusion in users [11].

## Results

All of our game modes proved to be fully functional. Including efficient methods of running the game, such as a generating the board GUI and array through a for loop; checking the entire board for a win.

The speed of the calculations also surprised us in the final draft of the system. With many recursive calls for the Player vs AI game and a huge amount of for loop iterations and method calls, we expected the system to sometimes respond late and not have an instant response. Nevertheless, the speed of the system was very fast and calculations for the AI and winning methods were worked out instantaneously.

Our programmed AI proved to be challenging upon observation of participants using the system and by our own use - during testing.

# Quality assurance

Quality assurance is the establishment of organizational management structures, procedures and standards which lead to high-quality products and/or services. There are several benefits to using a quality management system, such as; improved product quality and repeatability, increases processes efficiency and reduction in failure costs.

## Gantt Chart

In order to keep up with Quality Planning procedures, we decided to use a Gantt Chart to keep track of major tasks and their timings. A Gantt chart is an appropriate tool to use as it allows constant tailoring for large, complex projects. As such, we were able to use this Gantt chart to detail all major tasks and the majority of sub-tasks that needed to be complete and their time frame.

## Meeting Minutes

One of the methods we used as a form of Quality Management System is our detailed meeting minutes. Within our meeting minutes, each member of the group would detail what tasks they have completed from the previous meeting, and which tasks they want to complete by the next group meeting. These tasks would then be written up in detail in the ‘Action Item’ section of the meeting minutes - here, every member can see which tasks have been assigned to them and their due date. The benefit of this includes; improved product quality and repeatability and increased processes

efficiency. This is because by allocating each member tasks week by week, there is very little danger of overlap between the tasks done, and so as a group, we are able to complete. Furthermore, if a member is absent from a meeting, they are able to view the minutes from the meeting to keep up to date with project developments.

## GitHub

Another important part of ensuring the quality of the project is communication between the members of the group. We decided that we should use a medium which allows members to talk to each other and also upload/send pieces of work through the medium. Our use of GitHub ensured that members could access updated work and upload a version of work through the repository. The repository displays uploaded work, a description of what was uploaded, by whom and when it was uploaded. This allows other members of the group to view the most recent upload and continue working from there, as the description provides enough information to allow other members to build on it. This ensures quality assurance as work is not overlapped or modified by mistake.

## WhatsApp

Another important part of ensuring the quality of the project is to make sure there are reliable communication services between the group. In order to establish this, every member involved in the project was invited into a WhatsApp group, an instant-messaging and imagesending site whereby group members were able to discuss meeting dates and other important topics. This ensured every member was aware and attendant of meetings. As a result, there was no work delayed because of miscommunication.

# Evaluation

## Self-evaluation of product

Originally the game board was an HTML table which, although it functioned correctly, displayed a very blocky and unattractive interface. Therefore, we made use of action listeners and

‘onclick’ HTML attributes on ‘td’ elements and changed our board to just be a table. On testing, the new board looked more user-friendly and stylish.

Whilst observing participants playing our Gomoku game at the poster fair, it was drawn to our attention, that the characters were hard to distinguish between each player; due to the ‘X’ or

‘O’ style’s being the same text size, font and colour. Subsequently, once the poster fair was over a simple change was implemented into the code to assign an ‘X’ to be the colour green and an ‘O’ to be the colour orange. In order to stay consistent with the colour codes we used on our poster, these colours were originally red for ‘X’ and orange for ‘O’. However, once implemented the colours were hard to distinguish due to the similar colours, so green was chosen for

‘X’.

An issue we had with the design of the table was that the cells kept resizing when the text inside them was changed. After many attempts to fix this error, such as giving the table a fixed position, defining the height and width of the cells; the best solution we came up with was to make the size of the text inside of the cell smaller. This took from the style of the table, however, it hugely minimised the resizing effect that table had.

## Participants feedback of the product

One of the suggestions to our system was that the AI player is not able to be beaten. This is due to the way in which the algorithm has been implemented. But also due to the game of Tic Tac Toe not being able to be beaten if the player always goes first. As a result of this, future developments of the system would begin with the starting player switching between the user and the AI. Thus, the system would be fairer for the user as there would be more of a chance for the user to win.

Overall, we received positive feedback from participants (at the poster fair and amongst our computer scientist companions) about the functionality and the design of our games. In particular, one user complimented our layout and customisation of the website; thus, proving our consistent design efforts to be admired by users.

On observation, one of the problems which participants had issues with was changing the game mode from tic tac toe to Gomoku. We believe this is because we customize the buttons to be blue, therefore they did not resemble the usual form of an HTML button. As a result, we think the participants, on first glance, may not have noticed there was also a Gomoku game mode.

Consequently, it would be in our interest in future developments of the design to make the buttons more identifiable to the users.

Due to our extensive use of consistent designs, we purposely designed our tables to look the same. This was done with the purpose of keeping the same design to make it familiar and easy to use for the user. However, due to the tables looking exactly the same, we believe, participants were sometimes confused as to whether they were playing the player vs player or player vs AI game mode. Although we have a title at the top to address which game it is, these games could be distinguished by using a certain colour code or information to make it clearer to the user.

## Evaluation of repository services

Initially, the group repository was located on a OneDrive folder, whereby each member would download files onto their computer, develop them and re-upload them to the OneDrive repository. In the early stages this worked well - however, once we entered the development stage of the project, it was too difficult to work on the same file simultaneously as the repository would not update automatically to show new changes. Due to this, we decided to move our repository to GitHub. GitHub allowed us to work on the same file simultaneously - this is because when completing a milestone, we were able to upload the file to the repository with a description detailing the changes, thus allowing the second person working on the file to understand the differences in the code with ease.

## Future developments

As mentioned before in the participant's feedback of our product (4.2) an improvement to our system would allow the computer to take their turn first. This change could be implemented in alternative games so that in the first game the player takes their turn, then the second game, the computer takes their turn. Or else, the winner of the previous game could start the game.

It came to our attention that not many of the users were aware of the Gomoku game. As mentioned in 4.2, we believe this is because the buttons are not clear on the interface. To improve this issue, as a team we could experiment with different styles of buttons to determine which one is more eye-catching and less strenuous for the user to operate.

Our current solution uses a brute force

approach where it’ll check every possible move to an end state we have provided. This is one of the reasons we couldn’t make an AI for the Gomoku as there are too many possible end states for the game. In the future, we would like to have it actually learning while it plays. We think this could be achievable with the use of a neural network combined with the Monte Carlo tree search and also use the Minimax algorithm to update the weights of the nodes.

We attempted to make an AI for this game however it played very poorly. We used the Minimax with a new type of evaluation algorithm compared to the brute force approach and then combined this with the alpha-beta pruning optimisation algorithm. However, we still could not get it to play well. We think that if we could have got it to learn, as stated above, we could most likely get this to work. As only a few variables would need to be changed, such as the size of the board and the win count.

# Conclusion

To conclude our project, we are overly satisfied with the product we produced and the process of creating it. Throughout the project we were faced with challenges, some of them harder to overcome than others; however, we proved ourselves and created a rewarding final product.

Ultimately, we successfully managed to

create a functioning player vs player Tic Tac Toe and Gomoku game modes. As well as, implementing an AI for Tic Tac Toe where it picked the next best move possible and makes a hard challenge for the user, winning the majority of the time.

We created our games with stylish and user-friendly interfaces, which were then added to a single webpage. After making self-evaluations the product, we are happy with the final functionality of all the game modes, as well as the consistent and attractive design. The

website’s interface proved to be popular and easily operated by the participants who we tested the system on and the participants that tried our system at the poster fair. The website was also very popular amongst users; with some people repeatedly returning to our stall to try to beat the AI and challenging their companions to the game of Gomoku.

There are already a lot of Tic Tac Toe/Gomoku games available via the internet. We used these games as inspiration for our game and mainly drew on the limitations of these systems; so that we could learn from their mistakes. In terms of the AI games, some systems used different difficulties (ranging from easy to hard) in which the user could challenge themselves against. Our AI currently only has one difficulty which is hard, and we would like to implement this attribute in the future versions of the game - with the difference being that our system learns how to play from the user using a combination of neural networks and the Monte Carlo tree search.

Our website differs from competitors, as we noticed, from our research, that other applications have been limited to only a board and a notice of who won the game. It was our goal to create a website that gave each of the games more substance by playing vs the computer, playing vs their friends, playing Gomoku and recording scores in a pleasant easy to use interface. As well as, giving suitable information about the games on the website and provide a

recognisable logo, name and design which users can be loyal to.

With others trying to accomplish similar aims to ours we would recommend that other people developing a similar project, plan and research well ahead of time. Especially when it comes to establishing which algorithm will be used and what programming language you intend to develop in. As there are similar projects available they do offer pseudo code in different languages which can be developed into their own.

The AlphaGo project they have recently introduced, in the latest version of their project, AlphaGo Zero which was able to beat the previous version of the game (as mentioned in 1.2.1. DeepMind’s AlphaGo). The previous version the 18time world champion of Go; with the new development, the project is now able to defeat the number one champion. The AlphaGo project has been very successful and has shown the AI community that there is always room for improvement. We intend to use this as inspiration for when we improve Machine Learning XO.

# Acknowledgments

We would like to use this section to thank our project supervisor Radu Grigore for his persistent efforts and assistance to us during this project; as well as the University of Kent for providing the resources and means to complete the project.

# Bibliography

Techopedia, 2017. Artificial Neural Network (ANN). [online] Available at:

[<https://www.techopedia.com/definition/ 5967/artificial-neural-network-ann>](https://www.techopedia.com/definition/5967/artificial-neural-network-ann) [Accessed 10th Febuary 2018].

1. Explainthatstuff, 2018. Neural Networks. [online] Available at: [<http://www.explainthatstuff.com/introdu ction-to-neural-networks.html>](http://www.explainthatstuff.com/introduction-to-neural-networks.html) [Accessed 20th January 2018].

1. Tutorialspoint, 2018. Data Structures and Algorithms. [online] Available at: [<https://www.tutorialspoint.com/data\_str uctures\_algorithms/tree\_data\_structure. htm>](https://www.tutorialspoint.com/data_structures_algorithms/tree_data_structure.htm) [Accessed 1st February 2018].

1. Aaron Roth, 2017. *Algorithmic Game Theory*. [pdf] Aaron Roth. Available at: [<http://www.cis.upenn.edu/~aaroth/cour ses/slides/agt17/lect06.pdf>](http://www.cis.upenn.edu/~aaroth/courses/slides/agt17/lect06.pdf) [Accessed 29th February 2018].

1. G.M.J.B. Chaslot; M.H.M. Winands;

J.W.H.M; H.J. van der Herik; B.Bouzy

(2008). *Progressive Strategies for Monte-Carlo Tree Search*. [pdf] Faculty of Humanities and Sciences, University Maastricht. Available at:

[<https://dke.maastrichtuniversity.nl/m.wi nands/documents/pMCTS.pdf>](https://dke.maastrichtuniversity.nl/m.winands/documents/pMCTS.pdf)

[Accessed 15th February 2018].

1. Jeff Brady, 2015. Introduction to Monte Carlo Tree Search. [pdf] Jeff Bradberry. Available at:

[<https://jeffbradberry.com/posts/2015/09 /intro-to-monte-carlo-tree-search/>](https://jeffbradberry.com/posts/2015/09/intro-to-monte-carlo-tree-search/) [Accessed 1st March 2018].

1. Christopher, P., Jan, P., 2009.

Understanding Cryptography: A Textbook for Students and Practitioners.

Berlin: Springer-Verlag Berlin Heidelberg

1. GeeksforGeeks, 2017. Minimax

Algorithm in Game Theory | Set 4 (Alpha-Beta Pruning). [online] Available at:

[<https://www.geeksforgeeks.org/minima x-algorithm-in-game-theory-set-4-alphabeta-pruning/>](https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/) [Accessed 28th February

2018]

1. Cameron Brown, 2012. *Monte Carlo Tree Search*. [pdf] Imperial College London. Available at: [<http://ccg.doc.gold.ac.uk/ccg\_old/teachi ng/ludic\_computing/ludic16.pdf>](http://ccg.doc.gold.ac.uk/ccg_old/teaching/ludic_computing/ludic16.pdf) [Accessed 18th February 2018].

1. MAAS Digital, Dan Maas, 2016. *How AlphaGo Works*. [online] Available at:

[<https://www.dcine.com/2016/01/28/alph ago/>](https://www.dcine.com/2016/01/28/alphago/) [Accessed 4th February 2018].

1. CareerFoundry, 2016. *The Importance Of Consistency In UI Design* [Online]. Available from: [<https://careerfoundry.com/en/blog/uidesign/the-importance-of-consistency-inui-design/>](https://careerfoundry.com/en/blog/ui-design/the-importance-of-consistency-in-ui-design/) [Accessed 20th March 2018] [12] Encyclopedia Britannica, 2016. John

Vou Neuman, by William Poundstone [Online]. Available from:

[<https://www.britannica.com/biography/John](https://www.britannica.com/biography/John-von-Neumann)

[-von-Neumann>](https://www.britannica.com/biography/John-von-Neumann)

[Accessed 15th March 2018 ]

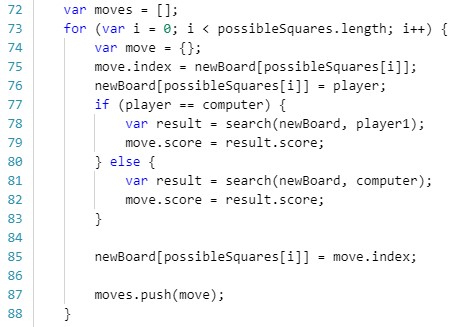
1. DeepMind.2018. AlphaGo | DeepMind.[Online]. Available at:

[<https://deepmind.com/research/alphago/.](https://deepmind.com/research/alphago/)

[Accessed 20 March 2018].

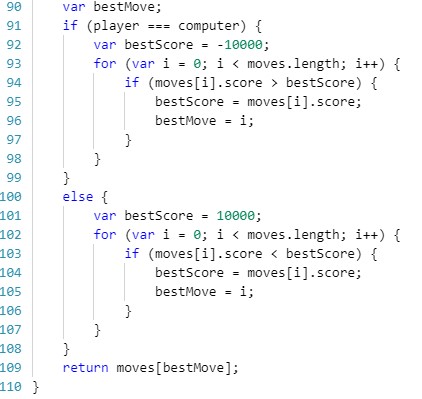
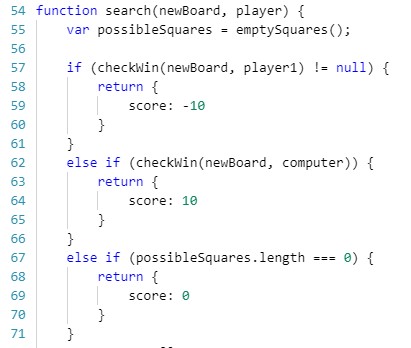
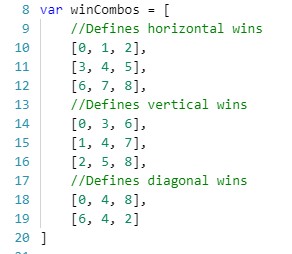
1. Machine Learnings. 2018.

*Understanding AlphaGo – Machine*

*Learnings*. [ONLINE] Available at:

*Figure A.1*

*Figure A.4*



[https://machinelearnings.co/understandi ng-alphago-948607845bb1.](https://machinelearnings.co/understanding-alphago-948607845bb1) [Accessed 29 March 2018.

# Appendix

*Figure A.3*

*Figure A.2*