

Connect 4 With Machine Learning

<Date>

Table of Contents

[Analysis 3](#_Toc54169838)

[Introduction 3](#_Toc54169839)

[End User Identification 3](#_Toc54169840)

[Questionnaire 4](#_Toc54169841)

[Proposed Idea 5](#_Toc54169842)

[Identification of Programming Language 5](#_Toc54169843)

[Numbered Objectives 6](#_Toc54169844)

[Documented Design 12](#_Toc54169845)

[Technical Solution 13](#_Toc54169846)

[Testing 14](#_Toc54169847)

[Evaluation 15](#_Toc54169848)

# Analysis

## Introduction

The problem area that I am interested in is Machine Learning[[1]](#footnote-1) which is a large topic in computer science research. I am planning on creating a machine learning algorithm which will train a neural network[[2]](#footnote-2) to act like a human at playing the board game, Connect 4. The main reason I chose this was because I watched a video on machine learning[[3]](#footnote-3) and was instantly mesmerized. I decided that I wanted to learn how to make something similar and chose Connect 4 as the game because it is quite simple and can demonstrate a learning AI quite well.

Machine learning algorithms have been around since the 1950s but has only recently taken off due to the rise of the internet and masses of data being transferred daily. Companies such as Google and Facebook are using machine learning algorithms daily to provide a better user experience, better than any human could do on their own. Many tools have been made to allow machine learning to be used by many people without much background in the topic itself. One of my inspirations is Google’s DeepMind[[4]](#footnote-4) AI which taught itself to beat a world-champion at the Chinese game of Go[[5]](#footnote-5).

Machine learning is based on mathematical ideas to adjust parameters such that a cost function is minimized. In my case of Connect 4, the cost function will be how different the AI’s choice is to the human’s choice so that when training, the AI will minimize this and start imitating the human better. I will derive all mathematics behind the algorithm as it is quite complex. The implementation is simple but, can get quite complex for efficient code due to a few tricks in organising the data in matrices and using matrix math to do calculations.

I am planning on only using a single Machine Learning algorithm for my final product as the number of tweaks to parameters I can make to the algorithm is massive. I should be able to adjust the algorithm so that I can get the most performance out of the single algorithm. If I finish the algorithm and realize that I there aren’t enough parameters to tweak, I can add another algorithm but this will add a lot of complexity and take a lot of time as a large rework will have to be made. This is because I would have to allow a modular kind of program where the number of algorithms available is not hardcoded, but more can be added at any time.

## End User Identification

As this project is meant to be used by multiple people, I have asked both my friend, Max, and my Computer Science Teacher, Mr Willans, to be my end users. Both have studied Computer Science and understand what is required to make a program. Therefore, asking them to be my end user will allow them to choose the right features to be put into my project without over complicating certain tasks.

After speaking with both end users, I have found that they have had no interaction with Machine Learning before and would not know where to start if they wanted to. This will allow me to create a program to demonstrate the power of machine learning and test, if enough is demonstrated, on someone new to the topic.

## Questionnaire

|  |  |
| --- | --- |
| **Question** | |
| **Max’s Response** | **Mr Willan’s Response** |
| Who are you, what do you do and are you interested in machine learning? | |
| My name is Max, I am a student at Ousedale School, and I am interested in Machine Learning. | Martin Willans, Head of Computing and Digital Applications at Ousedale School. I am interested in machine learning because it demonstrates the development of artificial intelligence in computing |
| What do you plan on getting out of using this project? | |
| To understand how an AI can learn using Machine Learning to play Connect 4 | I would like to be able to use the project to demonstrate artificial intelligence to students and use it as a simple and fun game |
| What features are you looking for? | |
| An easy to understand representation of how the AI is progressing | I would like some visual demonstration of the data that is being updated to see the machine learning algorithm working |
| Should the program be interactive or passive (the user only plays the game, and everything works behind the scenes)? | |
| The program should be passive as I am only interested in seeing the AI develop over time | I think that the user should play the game and the rest of the process should happen I the background |
| How do you want to interact with the program? | |
| I would like to be able to interact through a graphical interface with buttons and text boxes | There should be a visual element so that the player can place the piece in the right column. You should also be able to log in and export your data to a file to be transferred to another computer |
| How should the program look? | |
| I am not interested in too much detail in the graphical interface. Just enough so that it looks good but can also express all the information needed | Only simple graphics are required to show the gameplay also, the machine learning is the most important aspect, so graphics are not that important |
| Do you want the program to run on multiple operating systems e.g. Windows and Mac? | |
| I only own a Windows PC so would not benefit from having the program run on Mac | The program only needs to be able to be run on a Windows PC |
| Should the user account be transferable such that they can use their account on another computer? If so, elaborate | |
| I am only interested in playing Connect 4 on my own computer but would be nice to be able to play against someone else’s AI without having to train from scratch | It would be good if the user data were stored in a file that could be exported and moved to another computer to be imported and played with on that computer |
| What quality of life additions would you like to see in the project? | |
| An easy to use interface, multiple accounts so I can play against different AI’s I have trained, and be able to watch an AI vs AI game | I would like to see some data being saved about the user such as number of games played, and games won |
| Any extra words? | |
| N/A | N/A |

## Proposed Idea

My final plan is to create a Connect 4 application that can demonstrate Machine Learning in an interactive way. I will do this by allowing each user to create an account and be given an AI to ‘teach’. As the user plays Connect 4 against another player or AI, the AI that the user ‘owns’ will learn from the game being played and eventually begin to imitate the user. This imitation is personal to each user and allows users to relate more to the machine learning application remain engrossed in the program.

## Identification of Programming Language

Research about programming languages came from Ignite Digital[[6]](#footnote-6) and Raygun[[7]](#footnote-7).

* **Python** – Python is a very user-friendly language with many libraries that help development. It does supports Object Oriented Programming and has a lot of Quality of Life features built into it however, it is not the most performant language when coding at a basic level. I have a small amount of experience in Python but will have to research how it handles object-oriented programming.
* **Java** – Java, like Python, has an abundance of open-source libraries useful for certain applications (including Machine Learning). Again, it is not the most performant language but, has much more control over lower level ideas than Python does. I do not have much experience at all in Java, but the syntax is extremely like C/C++ meaning it will not take much to learn.
* **C/C++** – Both C and C++ are extremely low-level languages allowing for deep control of hardware to gain large performance boosts. C++ comes with its own libraries (STL[[8]](#footnote-8) – Standard Template Library) which are highly performant and almost never ignored for another library. Only C++ is Object Oriented out of the two meaning C will require a bit more work for certain areas. I am extremely familiar with C++ as I use this language for almost all my projects.
* **C#** – C# is an exceedingly popular language for development as it allows for low-level control like in C but also is much more user friendly like Python. C# is object oriented which is useful and certain programs such as Unity use C# as their language of choice meaning Graphics with C# is a lot easier than with other languages. I have almost no experience with C#, but the syntax is similar to C/C++ with simple pieces of code but changes a lot with function definitions and similar ideas.
* **Visual Basic** – Visual Basic is a simple and pseudocode like language meaning it is very user friendly with how it works. However, this causes some problems when it comes to consistency as some differences from other languages become apparent when using arrays as many times Visual Basic handles these differently and *off-by-one[[9]](#footnote-9)* errors occur are quite common. Visual Basic is the language we have been learning at School since year 10 meaning, I am quite familiar with it but due to its awkward Object-Oriented Programming it will require some learning.

In conclusion, after researching different programming languages, I have decided to use C++. This is because of its low-level control and great Object-Oriented Programming. The only problem with C++ is the use of Graphical Libraries, this is because as C++ is a language with a lot of control, the libraries it comes with also contain a lot of complexity meaning it is difficult to use them to do simple things. I would have chosen Python due to its user-friendly syntax however, I do not have a good enough understanding of how the Python compiler works in terms of multiple files to be able to create a large project with it.

## Numbered Objectives

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Main Objective | | Explanation | Success Criteria | | Sub-Success Criteria | |
|  | Create an AI that can learn from experience from human actions | A human will complete a task and the AI will try to replicate that task as best it can. As the human does more tasks the AI needs to learn to do all tasks like the human |  | A working neural network must be made |  | Be able to feed in an array of floating-point values and receive an array of floating-point values as an output |
|  | The neural network must have a runtime defined number of layers and number of neurons in each layer |
|  | Be able to apply different activation functions to different layers |
|  |  | Allow the initial values for weights and biases to be chosen with purpose |
|  | The neural network must be standalone meaning it has no relation to Connect 4. It is only used for that purpose |
|  | A machine learning algorithm must be in place to allow the neural network to adjust its weights and biases to learn from experience |  | Training data must be collected and stored so that the neural network can learn from it later |
|  | A cost function must be made to measure the deviation the neural network is from the human |
|  | A gradient descent algorithm must produce a change to all weights and biases from a single training example to minimise the cost function |
|  | An average over all training examples for changes to weights and biases must be calculated and applied to the neural network. |
|  | Training data must be stored in a file so that the neural network can minimise the cost function based on these training examples multiple times |  | A single training example consists of the inputs given to the network and the expected outputs |
|  | An entire Connect 4 game is built up from many training examples which all must be stored in a file |
|  | Multiple Connect 4 games must be stored to increase the number of training examples the neural network has to learn from. The number of Connect 4 games must be variable |
|  | The training data must be stored in a binary format so that it can be reconstructed and used even after the program has been closed |
|  | The neural network needs to be serialised and stored in a file so that it can be used when the program is run again |  | The number of layers and number of neurons in each layer must be stored in a file |
|  | The weight matrices and bias vectors must be stored in the same file as the size of the neural network |
|  | All information about which activation functions are being used must be stored in a file |
|  | The neural network must be able to be reconstructed from the serialised data and produce the same outputs as before being serialised |
|  | Create a working Connect 4 game | The Connect 4 game will take user input and display the current state of the board. It will also determine whether the game has been won and by who |  | The board needs to be displayed on the screen |  | The program needs to switch to a Connect 4 state which contains all instructions about the game |
|  | An empty Connect 4 board needs to be displayed as an image |
|  | Counters need to be displayed in the right positions on the board with the right colour |
|  | Both users’ names need to be displayed and which players turn it is currently |
|  | The users need to be able to place a counter in a specific column |  | Only the user whose turn it is currently should be able to place a counter |
|  | The user should be able to select a column by clicking on it and have a counter be placed in the lowest available place in that column |
|  | The game should check whether a user has won, or the game has been drawn |  | After each counter has been placed, a check should be done to see if four counters in a row have the same colour |
|  | If the game has been won, a game win state should be selected and displayed |
|  | A check should be done to see if the game has been drawn |
|  | If the game has been drawn, a game drawn state should be selected and displayed |
|  | Create a working account system | The user will have to create an account with details about them and this account will contain the AI that learns from that user |  | The user must be able to create an account |  | The user should input a username and password |
|  | If an account already exists with the inputted username then it should ask the user for a username again |
|  | The account should be created with a new AI and all details related to the user |
|  | The user should be able to delete their account |  | The user should be asked to input a username and check if that user exists |
|  | If the user exists, a password should be requested and if that exists then the account needs to be deleted |
|  | When deleting the account, the file containing the information needs to be removed |
|  | Each user should be able to access their account to play Connect 4 |  | When using an account both the username and password must match an already existing one |
|  | All data should be loaded into an object so that the AI can be utilised to either play the game or learn from one |
|  | The account should be able to be saved to a file |  | The username should be stored in the file and will be used as the name of the file itself |
|  | The password should be encrypted and stored in the same file as the username |
|  | All data about Connect 4 should be stored in the file such as number of Wins/Losses |
|  | The AI’s neural network needs to be stored in the file in a format such that it can be reconstructed |
|  | Create a pleasing graphical interface | Allow the user to interact with the program through a graphical interface using buttons and text boxes |  | Create a modular system where objects can be created and placed on a window |  | Allow the creation of an object such as a button or image and place it anywhere on the current screen |
|  | Allow the user to use the objects created to interact with the program such as change state to the game state |
|  | Be able to load in different resources such as textures to change the look of the program at runtime |  | Allow an image file to be changed and have the program reflect this change by using that image file |
|  | Load files from a folder which are used to describe how the program is to be laid out |
|  | Allow the user to view a leader board | Create a state where all users are displayed along with their total wins |  | Display all users in a list and allow them to be clicked on to view more information |  | Sort all users by a heuristic chosen by the user including alphabetical, number of wins, number of losses |
|  | When a user’s name is clicked, display all information about that user |
|  | When first using the leader board, all users must be loaded from their files and put into account objects |
|  | Leader board account objects should be stripped down as to save memory such as not including the AI itself |
|  | Allow the user to search for a username |  | Use algorithms such as Levenshtein Distance[[10]](#footnote-10) or Soundex[[11]](#footnote-11) to match similar usernames |
|  | Sort the list of users alphabetically when searching as usually the user will want to find one specific user |

## Prototype

I decided to prototype this project to make sure that it is simple enough to demonstrate Machine Learning quickly to keep the user entertained. I have chosen to remove a few features that are unnecessary for this task such as a neat graphical interface and leader board. This prototype will also be using an external library to speed up the process of development as it is just to allow the user to watch a learning algorithm take place.

The main take away from this prototype was the massive help that creating a state system had on development. This state system allowed me to create states and push them onto a stack which was used to determine where the user is in the program. This was also an easy way to deconstruct the program into sections and finish those programs before starting on others and see that part work through the state system. I will almost definitely be using a similar state system in the final project with some revisions to fix a few problems I had with it.



This is the entry point into my program which demonstrates how the state system should work. A new main menu state is created which is push onto the top of the state stack. Then a loop is started which runs until all states are removed. This would only happen if the ‘exit’ option were selected from the main menu state. If a state does exist on the stack, it is run which uses polymorphism[[12]](#footnote-12) to change what happens depending on what is currently the top state.



This is the definition of the ‘Neural Network’ class. Each user contains a single Neural Network[[13]](#footnote-13) and is only able to interface with it through its constructors and three methods: Feed Forward, Train and Serialize.



This Feed Forward method is a simple, vectorised implementation of forward propagation which is how data is manipulated in a neural network. The ‘Eigen’ keywords used here are to do with the Matrix library I am using for this prototype which allows easy Matrix-Math to be performed such as demonstrated with the line , which is a matrix-vector multiplication operation followed by a vector-vector addition operation.

I will be creating my own matrix math library in the final program to reduce the size of the executable because I only use a tiny amount of this library.



The Train method is where the updating of all weights and biases occurs depending on the output from the backpropagation method.



This is the program running on the main menu state. If the user inputs one of the options shown above, the main menu state will interpret this input and either push a new state depending on what was selected or pop the main menu state off if ‘exit’ was chosen. This causes no more states to be on the stack and the program to exit.



If ‘Play’ is selected on the main menu state, the user is asked to input details about who is playing the game including if they want the human or the AI to play.



The game is then repeated until someone wins. After the game is over, the AI will update using the previous game and execute the learning algorithm. This causes the AI to slowly change how it plays over time.

This prototype has shown me that the learning algorithm I am using requires a lot of training data to work effectively. In this prototype I am only using the previous game which is on average 24 moves per game[[14]](#footnote-14). This is no where near enough training data for the algorithm to learn quickly. After many games with the AI, I did find that it was changing very slightly but was mainly choosing the same column the entire game until it was full. This suggests that the neural network is not effectively using the input data it is given and just producing the same results each time.

Due to this problem in the learning algorithm, I have decided to save the past couple of games in the file along with the data about the player. Then when I come to train the algorithm, it can use the new game just played and the previous games to reinforce what it has already seen. In a perfect world, I would be able to train on all training examples every time however, without a custom compression algorithm, storing all past games would take up a lot of storage and I would prefer it if I could keep the data related to the user as small as possible.

Even though this is a prototype, just seeing that the AI changes what it does after each game is really interesting and I believe that the final program can be extremely fun and engaging for new users.

# Documented Design

# Technical Solution

# Testing

# Evaluation

1. <https://en.wikipedia.org/wiki/Machine_learning> [↑](#footnote-ref-1)
2. <https://en.wikipedia.org/wiki/Neural_network> [↑](#footnote-ref-2)
3. <https://youtu.be/gn4nRCC9TwQ> [↑](#footnote-ref-3)
4. [https://deepmind.com](file:///C:\Users\josep\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\LocalCache\Roaming\Microsoft\Word\.git) [↑](#footnote-ref-4)
5. <https://en.wikipedia.org/wiki/Go_(game)> [↑](#footnote-ref-5)
6. <https://www.ignite.digital/10-best-programming-languages-to-learn-in-2020/> [↑](#footnote-ref-6)
7. <https://raygun.com/blog/programming-languages/> [↑](#footnote-ref-7)
8. <https://en.wikipedia.org/wiki/Standard_Template_Library> [↑](#footnote-ref-8)
9. <https://en.wikipedia.org/wiki/Off-by-one_error> [↑](#footnote-ref-9)
10. <https://en.wikipedia.org/wiki/Levenshtein_distance> [↑](#footnote-ref-10)
11. <https://en.wikipedia.org/wiki/Soundex> [↑](#footnote-ref-11)
12. <https://www.tutorialspoint.com/java/java_polymorphism> [↑](#footnote-ref-12)
13. <https://www.investopedia.com/terms/n/neuralnetwork.asp> [↑](#footnote-ref-13)
14. <https://www.reddit.com/r/math/comments/1lo4od/how_many_games_of_connect4_there_are/> [↑](#footnote-ref-14)