

Connect 4 With Machine Learning

<Date>

Table of Contents

[Analysis 3](#_Toc53669690)

[Introduction 3](#_Toc53669691)

[Identification of Programming Language 3](#_Toc53669692)

[Numbered Objectives 4](#_Toc53669693)

[Documented Design 9](#_Toc53669694)

[Technical Solution 10](#_Toc53669695)

[Testing 11](#_Toc53669696)

[Evaluation 12](#_Toc53669697)

# Analysis

## Introduction

The problem area that I am interested in is Machine Learning which is a large topic in computer science research. I am planning on creating a machine learning algorithm which will train a neural network 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[[1]](#footnote-1) 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[[2]](#footnote-2) AI which taught itself to beat a world-champion at the Chinese game of Go[[3]](#footnote-3).

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.

## Identification of Programming Language

Research about programming languages came from Ignite Digital[[4]](#footnote-4) and Raygun[[5]](#footnote-5).

* **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[[6]](#footnote-6) – 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[[7]](#footnote-7)* 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 |

# Documented Design

# Technical Solution

# Testing

# Evaluation

1. <https://youtu.be/gn4nRCC9TwQ> [↑](#footnote-ref-1)
2. [https://deepmind.com](.git) [↑](#footnote-ref-2)
3. <https://en.wikipedia.org/wiki/Go_(game)> [↑](#footnote-ref-3)
4. <https://www.ignite.digital/10-best-programming-languages-to-learn-in-2020/> [↑](#footnote-ref-4)
5. <https://raygun.com/blog/programming-languages/> [↑](#footnote-ref-5)
6. <https://en.wikipedia.org/wiki/Standard_Template_Library> [↑](#footnote-ref-6)
7. <https://en.wikipedia.org/wiki/Off-by-one_error> [↑](#footnote-ref-7)