Comparing the Effectiveness for Solving Optimal Paths using Machine Learning and Neural Networks Against More Traditional Algorithms

James Smith - 863266

***Abstract*— Over the last decade, we have seen a rise in the use of machine learning(ML) and neural networks for solving large-scale problems. [1] This is mostly due to the introduction of big data. Coming with this constant stream of information is the need to analyse it and take meaning from the numbers. Machine learning has been popular in this respect. Using ML in such field as medicine [2] it can be used to extract data efficiently and even provide [increasingly] accurate interpretations [over traditional approaches].**

I. INTRODUCTION

Over the course of this project, I aim to find areas where machine learning may not be the best approach. Analysing areas such as speed in problem-solving, efficiency and size. Pathfinding and route planning are both important parts of this AI based world we are slowly crafting around us. With the advent of self-driving cars and mobile robots, many may consider adopting an optimal route solver to get around efficiently. The problem with this constantly mobile world we live in is that optimal may not always stay optimal and new routes come and go. Can this be better put into practice using a machine learning approach? Using collected data to adapt to new environments on the fly and constantly learning from the changing surroundings? Firstly I will analyse current pathfinding techniques, exploring their strengths and weaknesses. Then I will expand my study onto machine learning comparing the two. As part of this, I will build a practical model to demonstrate the differences in a more visual format for later presentations. Route finding is already quite a well-documented topic. It has been an important part in developing current technologies such as in; navigation, video games, and even computer networking. [7] As far back as the 1800s mathematicians have worked on an early version of this problem, known as the travelling salesman [8] The thought experiment was based around finding the shortest paths between a collection of towns. This problem was then studied over the course of a few decades by many science researchers from all backgrounds including mathematics, computer science, chemistry, and physics. A solution would have benefitted many. A few approaches have been formulated ranging from basic brute force techniques leading onto more heuristic approaches *which can find solutions for extremely large problems (millions of cities) within a reasonable time which are with a high probability just 23% away from the optimal solution.* [9] In 1993 artificial intelligence researcher Marco Dorigo discussed a way for finding good' solutions by simulation an ant colony. Modelling the solution after the behaviour of ants finding the shortest path between food sources and their nest. This still only found good solutions. This brings us onto more widespread solutions such as the A* algorithm. Widely used in traversing graphs and pathfinding as is offers good performance and accuracy. It is a complete algorithm and as such will always find a solution if one exists. Another algorithm, conceived by Dutch computer scientist E. E Dijkstra in 1956 finds a solution following a different approach. I will discuss these solutions later in this document to analyse which would be best to use in this project for our traditional approach'

After looking into previous solutions I will now move onto the rival approach I intend to produce. One benefit of machine learning optimisation is that it gets better over time. Meaning, the more data supplied to the problem, the closer to a perfect solution it can get. My aim is to find weather with enough time can the learnt algorithm beat the time to find a correct solution over a past approach. Initially, there appear to be a few weakness in my approach. The time needed to train such a model may be far greater than the time I have been allotted. Also, the computing power required is also an issue. To train this model I may require some outside host to constantly run the simulation to train the data. There is also the worry that example program is too specific and not relative to the global message. Comparing two algorithms in this way may not yield fully accurate results. This allows room for the project to grow beyond what I am able to do in my time limit. In summary, this study aims to discuss the practicality of machine learning. Is it the breakthrough new technology, aided with the current advancement in large data collection, destined to grow into a world changing technology? Could it possibly be just an old mathematical method with interesting applications in high-speed data analysis? I predict it to be a mix of both, offering benefits in many areas of study. Just not the artificial intelligence solution that solves any problem we throw at it.

II. LITERATURE REVIEW

For a stronger starting point, I have taken the time to read up on similar projects. This will hopefully allow for a clean and efficient start. Cutting out the early problems I may encounter by taking solutions from other sources. As discussed later, time management is crucial. This includes not spending too long in getting a prototype up and running. The following sources detail similar problems or required reading to understand ideas I will be building upon.

A. End to End Learning for Self-Driving Cars[10]

This real-world application, while more focused on image recognition, shows a real-world application of the ideas discussed in this paper. The way the training dataset is built from scratch by performing the desired task is similar however as this not a simulation the Convolutional Neural Network (CNN) had to be trained from human performed tasks and set to accurately mimic their behaviour. A second interesting observation is the use of image processing to reduce the incoming data into the raw elements deemed useful to the neural net. If the whole image was supplied the time required to process the data set would be much larger. This is something to bear in mind when deciding what values to supply my own implementation of this network.

B. Steering Behaviors For Autonomous Characters[11]

This paper looks further into the uses and situations an autonomously controlled character would encounter. Actions such as leader following, evasion, pursuits and slowing down on arrival as to not force an abrupt deceleration simulating more natural driving techniques. These defining behaviours are all implemented through study and modelling. With the neural net approach I could hope to find similarities to these behaviours and with the right influences and data input, I may start to see if they can be learnt rather than simulated.

C. Autonomous characters in virtual environments[12]

While this paper discusses the area and surrounding topics in a lot more depth the main points of interest are sections; 2.1.8 Neural Nets, 2.3.6 Genetic Algorithms, 3.4.5 [Implementation of] Neural Nets & 5.3.4 More Neural networks. Throughout differences between pre-learning and active learning are discussed. Active training is to feed the trained data into the neural net as the program runs, meaning it is constantly learning and taking on new solutions. The other approach of pre-learning means supplying the data sample before running the program. This can be data from previous runs but not the current execution. The paper states that active learning is not ideal from a creator - consumer model as found in video games as once released the developers have no control in what is displayed on the screen. However, I will not be releasing this as a product and merely a simulation of two ideas which negates that argument. Then again, one consideration to make is that I will need to demonstrate this at a later date. Having a badly behaving bot is not something I wish to leave to chance and will be sticking to the pre-learning approach.

D. Game AI: Simulating Car Racing Game by Applying Pathfinding Algorithms[13]

Now that we have discussed neural nets and how other studies structure their design I will look into other more algorithmic approaches. This paper demonstrates the practicality of a modified A* algorithm to achieve a similar goal. As discussed earlier there are several approaches one can take when designing an algorithm to solve a pathfinding problem. The choice of algorithm is largely down to the type of problem and there is no one true solution. Modelling a famous speedway the image is then processed into a two-dimensional array of waypoints along the track. This is because the game world is needed to be represented in a manner that allows the search algorithm to run. The car must then find the shortest path between these pointers. These solutions will be useful when developing my own approach, however, I will not be placing any waypoints by hand. Whether or not the neural net can outperform this will be largely down to my optimisation of the initial algorithm. CPU cycles are to be kept to a minimum.

E. Optimization-based Motion Planning in Virtual Driving Scenarios with Application to Communicating Autonomous Vehicles[14]

One area I have not covered so far is the character/pointer design. What will the computer be controlling around this maze? I have several options ranging from a pixel that can move in any direction up to a fully modelled vehicle with steering controls. In this document, the mathematics behind steering and kinetic car modelling is discussed. If I were to use a model like this the implementation here could heavily influence my design and speed up the process. Systems like delayed control over velocity and axle based steering would create more of a realistic implementation. This paper then goes onto emulation car manoeuvres, another area that this project could adapt into at a later date.

III. Approaches

As Part of my introduction, I briefly covered which current practices are used to solve the route finding problem. After much research, I will now discuss these methods in more detail and hope to find which will be best for use.

A. Brute-force

Brute forcing a solution would be a good place to start. Its the process of attempting every possible combination of solutions until the optimal one is found. The problem here is that it will be unable to tell if we have found an optimal solution without trying every possibility. For smaller, simpler mazes the solution will be found quickly but as the problem grows in complexity the time taken will grow exponentially. There has been some refinement in this approach which I will now discuss

B. A*

A* (pronounced A-Star) stared its history with the goal of automating movement.[17] It was originally created as part of an attempt to build a mobile robot with the ability to plan its own paths. The main aim was to find the least cost paths for this robot to take, both in terms of processing power and distance. Thus a heavily optimised search algorithm was created. Typical implementations of the A* form of a priority queue. This finds us the shorted path by estimating the distance from the current location (node n) to the target location (goal node) giving us the heuristic function h(n). By keeping track of the distance from the starting point to the

current node we get g(n). These two values summed together give us

$$f(n) = g(n) + h(n).$$

A* selects the path that minimises f(n) where n is the next node on the path. A* terminates when the path it chooses to extend is a path from start to goal or if there are no paths eligible to be extended. The heuristic function is problem-specific. If the heuristic function is admissible, meaning that it never overestimates the actual cost to get to the goal, A* is guaranteed to return a least-cost path from start to goal.[17]

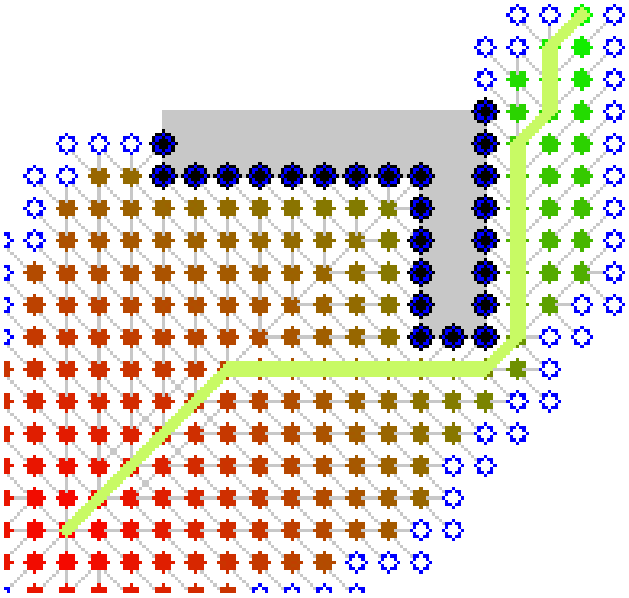


Fig. 1. Illustration of A* finding the shortest path using nodes

Using this shortest path data, a route can be created by modifying each node to keep track of the position of the previous node. This means a route from the goal node can be followed back to the starting node.

C. Any-angle path planning

This collection of similar algorithms search for a path between two points in space, allowing the turns to have any angle. This gives us a path with few turns and a much neater path than that of an A* approach which confines the route to a grid of nodes.

Four main any angle path planning algorithms, based on the original A* concept have been produced, all of which solve different specific problems including taking into account three-dimensions and creating interval nodes between fixed points to find increased accuracies from low-resolution grids

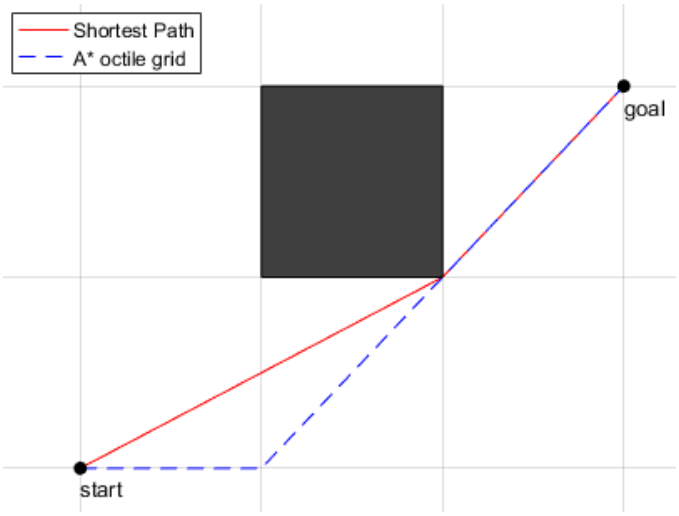


Fig. 2. Illustration of an Any angle path compared to A*

D. Dijkstra's algorithm

Rather than being based on a similar A* approach Dijkstra's Algorithm is closer to a spanning tree solution know as Prim's algorithm[15]. The shortest path tree is created and the source set as root. *This algorithm makes no attempt to direct "exploration" towards the destination as one might expect. Rather, the sole consideration in determining the next "current" intersection is its distance from the starting point. This algorithm, therefore, expands outward from the starting point, interactively considering every node that is closer in terms of shortest path distance until it reaches the destination. When understood in this way, it is clear how the algorithm necessarily finds the shortest path. However, it may also reveal one of the algorithm's weaknesses: its relative slowness in some topologies.*[16]

IV. Summary and Analysis of Chosen Approach

To sum up, the end goal of this project is to have developed a challenge to be solved using two competing techniques. Analysing both Reinforcement learning and an algorithmic approach. The task presented will have to differ enough each time so that the bot isn't only learning to solve a specific maze by following a set of movements but any maze through a more tactical approach. I will model these maps as a grid of filled or unfilled blocks. This will also feature a scalable resolution allowing for the complexity of the maps to differ and grow harder changing the time in which the algorithmic approach takes to find a perfect route. The neural web built to solve these maps will carry over to the next task, improving the efficiency each time. The aim here is to see how many iterations we must go through before we beat the algorithmic approach (if ever) As the initial bot will be starting out with learning to drive and navigating basic obstacles the initial maps will be very low resolution and difficulty. These will consist of basic shapes such as straight lines leading on to simple bends and curves. Initially, I predict the algorithmic approach to solve these instantly while the bot will be making

several attempts to even move forwards consistently.

A. Discussion of Programming Language choice

Choice of programming language will be significant in this project as I will be heavily relying on existing libraries and packages. I am already pretty certain on my choice of python, however, looking into other options may reveal either an alternative option or give me further insights into pythons weaknesses.

1) Python: I have always had in mind that this project would be best created using python. It is a strong language to build and deploy quick ideas. The key to this is its simplicity and the vast supply of packages and libraries. I am already quite familiar with python and have experimented with the OpenAI gym library. The only possible downside is ensuring it is stable enough to run for an extended period of time and record the training data. Some research into this has proven that there are quite simple solutions, however.

2) JavaScript: Javascript was another possibility to use. I have practised some linear regression algorithms in a web-based setting before. I also know it features well and would be more comfortable creating a UI. Tensorflow also would allow me to quickly set up a training environment and offers many solutions to the AI system. From day one this would need to be hosted on a server, even just as a localhost, to ensure it is running smoothly. This allows for an easier transition to another host for when the gathered data becomes too large to host locally. Also with it being web-based, I could then go on to distribute it much easier to a wider audience. Helping with the data gathering process

3) Java: Another approach to take would be using Java. More object-oriented I could track the hierarchy of my design more efficiently. Javadoc would also be a great help in keeping the codebase well documented. While I am proficient enough to create this in java, I have little experience with what AI packages are on offer and as such would need to do more research to be up to the same speed as I am with python.

4) Further considerations: These following languages are my weakest in terms of developing a full program but still worth discussing for their merits;

C++

Fast and efficient C++ would be a strong choice. It also offers the Tensorflow package (as was used to create Tensorflow in the first place) However I am only just starting out as a C++ developer and would spend most of my time learning the language rather than forming solutions.

R

R is a great tool for analysing large sets of data with optimisation built in mind. However, the data I will be producing is mostly visual. While it could offer an alternate solution to the data processing step, running and interacting with the model would prove tricky. It is something to consider in the future for other machine learning tools.

In closing, I have decided to stick with Python as it offers everything I require for a solid foundation for this project. Using the available libraries and packages I feel it will be the most efficient approach.

V. Structure

A. System Design and Implementation

After looking at the many different design implementations I could take with this project, I have decided upon one main approach in the form of maze solving. Comparing two different solutions to the same problem, I will take one current best practice, A* (as discussed in section 3.0) and implement it alongside a blind bot that will learn over time to solve any maze thrown at it. I predict it to do badly at first, struggling to even move in a straight line, but once left running for long enough should be able to navigate with ease. There are three main components of the design.

1) Maze Creation
2) Optimal Solution finding bot
3) Machine learning bot

Two copies of the mazes I generate will sit side by side, one with the optimal solution finding algorithm, the other with a bot controlled by a neural network. These will be run at the same time for a more visual representation of what I am comparing.

1) Output Specification and Design: Generating a maze to solve should be a simple process, taking from similar works I can create my own version that is specific to my needs. As one considerable factor will be maze complexity, I aim to add in a difficulty level to the maze creation engine that will change the size and number of unique paths for each problem. The optimal solution finding will also be straightforward as I will just be implementing an existing algorithm into my design. To achieve some semblance of fairness I will match the speed of both the bots. This means a win case' will not be achieved until the bot reaches the end of the maze, not just when it knows' the solution to the maze. It will still have to follow the path is draws itself. Finally, the machine learning bug. There are currently many libraries out there that implement the process for me. As discussed in the previous section I will be using python and its existing libraries to solve this problem. My main task will be visualising its workings for easy debugging and also what data to send as an input. Again I have considered these issues in the previous section and initially plan to only have distance sensors from the bot to the maze walls and using a distance from exit value as a metric of its maze solving ability, only available to the bot upon termination (as so not to rely on this for pathfinding, defeating the purpose)

B. Development models

In order to keep to a tight schedule, I intend to follow a development model. These are ways of approaching the creation of a program that keeps you up to date and on track. I will analyse several here and come to a conclusion for which would work best for me.

1) Waterfall: The waterfall development model is set out in phases. Each phase must be complete before the next one can begin. This can be easily diagrammed, which is where it derives its name as the illustration would resemble a waterfall. This model tends to lead to good documentation and is very client facing. Offering a good insight into how the project is progressing from a non-technological stance. This is a good choice when requirements are understood and the process will undergo little to no change over the course of its life cycle The downsides to this approach are usually down to the nature of the project. With my plan requiring a large amount of change to finalise an area of development a fixed approach may not be best. However, the time management it offers would be an attractive addition.

2) Spiral: Spiral development is useful for product life cycles. The workload follows a circular path from planning to pre-evaluation and risk analysis. This leads onto Development which in turn gives way to further planning of the next phase. The spiral comes from the visual of progress spiralling' towards the centre - project completion. This is more useful for large-scale project development with many versions and updates for a larger workforce.

3) Rapid Application Development: This method promotes speed and quick prototyping. Workshopping ideas and quickly finding solutions, reusing old components frequently and adapting on the fly. This is a good approach when working with a client who requires a quick turn around and fast bug fixing. As I will not be working for a client, I will know how the project is progressing and as such this approach is less suited.

4) Agile: For Agile development the workload is split into separate areas, each independent of another. This allows for rapid development when working as a group. A worker assigns them self a task and completes it independently. Once each section is complete the areas are integrated and a full product is realised. This allows for changes in requirement and continuous development. No one section is waiting for another to be complete. Documentation is usually less thorough in this approach as each working is operating under their own guidance. I feel this would be less of an issue when tasked to only myself. It also benefits me in being able to work little and often on this project. Short burst will be easier to manage around my studies over a week's solid work.

C. Summary

I have decided to work in a scrum like Agile model using the waterfall approach for larger sections (as outlined in the time management section with use of a Gantt Chart). Using agile development within these larger sections I will split my workload into independent chunks and work in short bursts, using a goal-oriented methodology.

VI. Risks

As I've covered previously, to achieve what I need in such a short space of time in need to manage my time very carefully. This means also avoiding any potential pitfalls. The prospect of losing even a week to poor planning could cause a serious knock-on effect. In this section, I will discuss various areas where I could lose time and progress. My first big hurdle will be working around other portions of my course. This includes other coursework and exams I will be working on during this timeframe. Taking this into account I will try and reduce planned workload around these dates to ensure everything remains to a satisfactory level. As for Exams, I have planned a week to two off from this project where I will focus solely on exam preparation. This will also give me time to reflect on current progress and come back to any problems with a fresh mind. Another would be any form of illness that affected my ability to concentrate on work. During winter the likelihood of catching something is increased. To prepare for this I will aim for deadlines a little earlier, giving myself breathing room for any delays in the workflow. Keeping fit and healthy will be of great benefit. Data loss is also a cause for concern, the loss of any work will result in time lost fixing the problem. I intend not only to have version management implemented through GitHub but also store my files in a cloud service such as google drive or Dropbox. Resources and ability. While I have planned this well to be within my capabilities, it is always a possibility I come across an area I know little about and will require extensive research. Bugs and error fixing will be a large part of getting a prototype off the ground. This will impact the speed of development meaning sufficient research is a must.

VII. Time Management

Date outline Deadline for Initial Document

- 29/10/2018

External (known) Deadline conflicts

- 5/11/18
- 12/11/18
- 19/11/18
- 03/12/18
- 10/12/18
- 08/01/19

Exam period

- 08/01/19 - 25/01/19

Gregynog

- 30/01/19

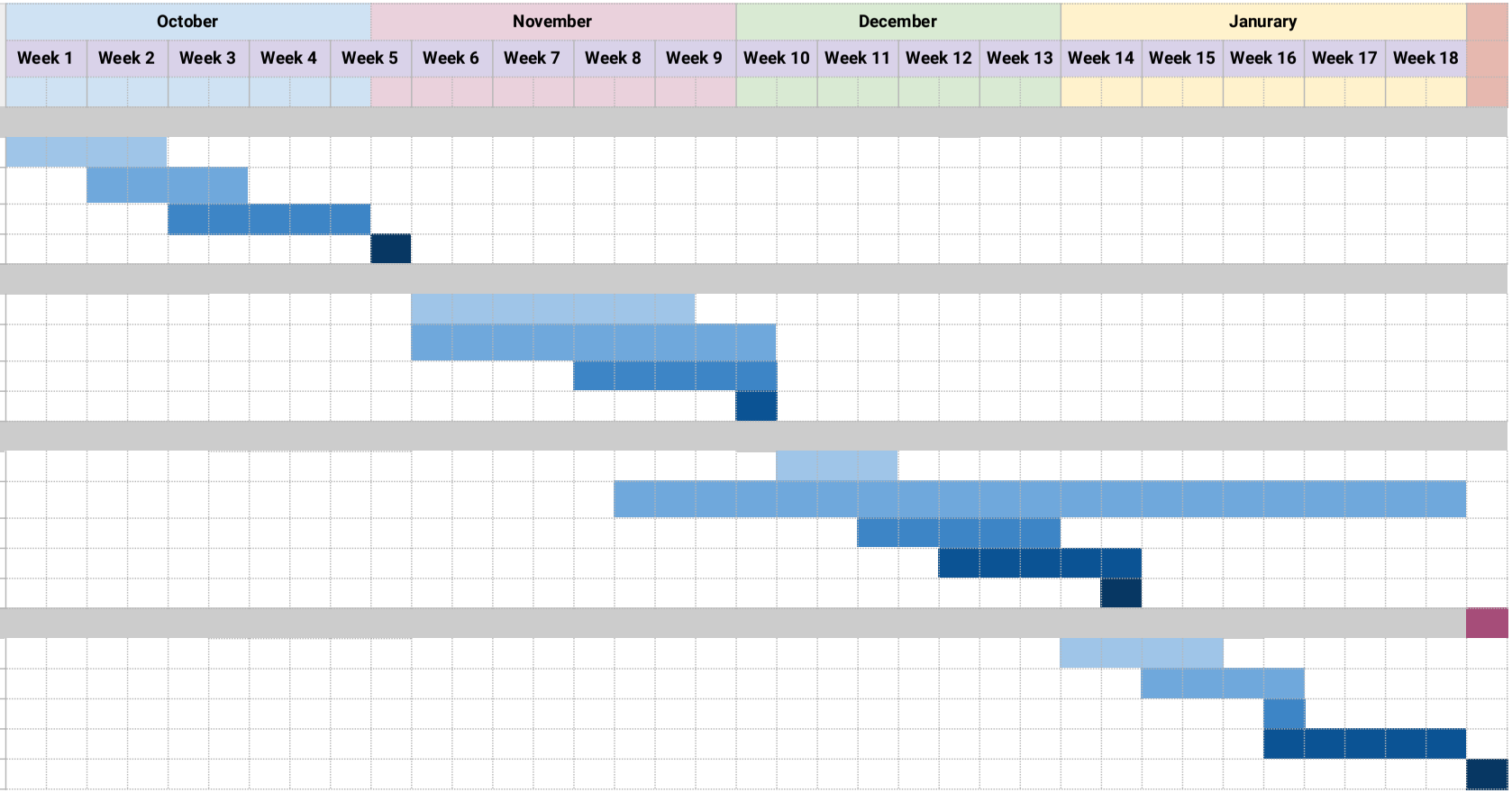
Project Fair

- ??/05/2019 - final date TBA

As discussed in the previous section, time management will be a crucial part of this project due to its short-term nature. With only 7-8 months part-time work the time allocated to important areas will need to be well thought out. On the next page is an outline of my workflow up until the Gregynog deadline. This is aimed to work around my other courses as well and leave enough time to create a project worthy of presentation. While the rehearsal period may seem extraneous, I have timed it to be running alongside my exam revision dates meaning it will be a slow burn over the two weeks.

Project Outline

TASK TITLE	START DATE	DUE DATE
Initial Proposal		
Initial Project Planning	1-10	13-10
Research	7-10	20-10
Write up	13-10	31-10
Initial Proposal Deadline	1-11	1-11
Project Definition and Planning		
Tech Research	8-11	23-11
Demoing Tech	8-11	5-12
initial concepts prototyping	23-11	6-12
working alpha model dl	6-11	7-12
Project Conception and Initiation		
model optimisation and consoliation	8-12	18-12
gathering training data	28-11	30-1
GUI build	10-11	30-12
beta model build	14-12	8-1
beta deadline	12-1	12-1
Gregynog		
Presentation planning	1-1	14-1
Presentation writeup	15-11	20-1
Presentation deadline	20-1	20-1
Presentation practice	17-1	30-1
Gregynog presentation	30-1	31-1



REFERENCES

[1] https://www.smartdatacollective.com/rise-of-machine-learning-ai-improving-lives/

[2] Jul 30, 2018 Rise Of The Machines: The Future Of Data Science And Machine by Meghann Chilcott Learning https://www.forbes.com/sites/forbestechcouncil/2018/07/30/rise-of-the-machines-the-future-of-data-science-and-machine-learning/#10de200e5009

[3] The Rise of Machine Learning in the Age of Data by Marius Banici https://www.3pillarglobal.com/insights/the-rise-of-machine-learning-in-the-age-of-data

[4] Computational Intelligence and Neuroscience Volume 2018, Article ID 7068349, 13 pages https://doi.org/10.1155/2018/7068349

[5] OI 10.1109/TPAMI.2016.2587642, IEEE http://personal.ie.cuhk.edu.hk/ pluo/pdf/ouyangZWpami16.pdf

[6] Doulamis, N. Multimed Tools Appl (2018) 77: 9651. https://doi.org/10.1007/s11042-017-5349-7

[7] https://www.khanacademy.org/computing/computer-science/algorithms/intro-to-algorithms/a/route-finding

[8] https://en.wikipedia.org/wiki/Travelling_salesman_problem

[9] Rego, Csar; Gamboa, Dorabela; Glover, Fred; Osterman, Colin (2011), "Traveling salesman problem heuristics: leading methods, implementations and latest advances", European Journal of Operational Research, 211 (3): 427441, doi:10.1016/j.ejor.2010.09.010, MR 2774420.

[10] End to End Learning for Self-Driving Cars arXiv:1604.07316v1 [cs.CV] 25 Apr 2016 https://arxiv.org/pdf/1604.07316.pdf

[11] Steering Behaviors For Autonomous Characters Craig W. Reynolds http://www.red3d.com/cwr/steer/gdc99/ http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.636&rep=rep1&type=pdf

[12] Wood, Oliver Edward(2005) Autonomous characters in virtual environments: The technologies involved in artificial life and their aspects on perceived intelligence and playability of computer games, Durham theses, Durham University. Available at Durham E-Theses Online: http://etheses.dur.ac.uk/2374/

[13] International Journal of Machine Learning and Computing, Vol. 2, No. 1, February 2012

[14] Optimization-based Motion Planning in Virtual Driving Scenarios with Application to Communicating Autonomous Vehicles Matthias Gerdts and Bjorn Martens arXiv:1801.07612v1 [math.OC] 23 Jan 2018 https://arxiv.org/pdf/1801.07612.pdf

[15] https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

[16] https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

[17] https://link.springer.com/chapter/10.1007%2F978-3-642-02094-0_7

[18] https://arongranberg.com/astar/

[19] Numerische Mathematlk 1,269-271 (1959) A Note on Two Problems in Connexion with Graphs by E.W.Dijkstra http://www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf