

Computer Games Development CW208

SRS and Project Report

Year IV

|  |  |
| --- | --- |
|  |  |
| Emmett Byrne  C00220950  Submitted: 3rd May 2020 | |

# 

Contents

[Project Abstract](#_rkvjwt77p0vm) **3**

[Project Introduction](#_2n3e0fhggxl6) **3**

[Background](#_hnertkdgvad4) **3**

[Project Description](#_79bsyr33800o) **4**

[Overview](#_zgoi84zdk2h5) **5**

[Philosophy](#_gbvp4p9lc3jf) 5

[Common Questions](#_57aq83vef7m4) 5

[Define the Application](#_gohw3lacj0ek) **6**

[What is the application supposed to do](#_pw1bma3gs694) **6**

[Experiment](#_rcdkug5bye2n) **7**

[Metrics](#_728ogw33w60p) **9**

[Success Rates:](#_mg7e2b55617j) 9

[Moves Taken:](#_jmpx30lmu0ny) 10

[Time taken to train and complete maze:](#_v3mzrad8rzgv) 11

[Results:](#_p6s2yu7nfl37) **12**

[Is there a precedent for this application?:](#_1b2dyw9v8w8r) **14**

[Project Milestones](#_1oshbvaixn6p) **15**

[Project Review and Conclusions](#_xz9oat7mkpym) **16**

[**References**](#_g9jbxw9nt9hq) **17**

# 

# Project Abstract

My project is to create a neural network and investigate its capabilities to solve a 2D maze with very limited information. The maze is built in Python and can be rendered out using PyGame and the Neural network is built with Tensorflow and uses Q learning a reinforcement learning algorithm to teach itself how to solve the maze. The maze is randomly generated and the amount of walls in the maze varies based on it’s difficulty level.

I created an experiment to test the Neural networks capabilities by getting it to train continuously on randomly generated mazes and increasing the difficulty over time until the maze is too difficult for it to complete. Once the experiment has finished we generate metrics and graphs and are able to come to a number of interested conclusions.

# Project Introduction

For my project I want to research a neural networks ability to complete a simple 2D obstacle course.The obstacle course I want to have the Neural networks complete is a simple procedurally generated 2D grid of different types of tiles where the network must move from tile to tile in order to reach its goal, made in C++ with SDL to render it. The objective of this is to create a data set from the results so that we can assess if we can create a neural network that can accurately complete the obstacle courses we give it.

# Background

Before starting this project I had practically no experience at all in working with Neural networks however I have always massive interest in making a Neural network as I have always enjoyed reading and watching sources of Neural networks solving interesting problems which is where I got the idea of creating a Neural network to solve a maze from since I've seen many examples of Neural networks solving mazes but I always found it strange that in each of these examples the Neural had perfect information about the maze and from this I decided it would be much more relevant to machine learning if the Neural network had very limited information about the maze.

All of the technology I used in this was completely new to me as I had never used either TensorFlow or Pygame. Fortunately I did have some limited experience with python having made some simple projects in the year prior to starting this project.

# Project Description

The obstacle course I will be creating will be a procedurally generated 2D grid of various different tiles. The character will start at the left of this grid and attempt to work across to the right of the grid. Obstacles that the Neural network will face will include impassable tiles that it can’t move into, tiles that impart a movement penalty to slow it down, tiles that will automatically move it in a specific direction and tiles that can change between the previously mentioned states. A simple algorithm will be implemented to ensure the course can be completed and in the case it can’t be completed then a new course will be generated. There will be a move limit that will make the neural network fail the course if it takes too many moves.

For the neural network we will be using a library called Tensorflow which uses python. With this we will create a model for our network which will consist of 15 input neurons, 12 of these input neurons will represent every tile within 2 spaces of the network in the obstacle course and the last 3 neuron will represent the current position of the network, the amount of moves it has remaining and the position of the goal. we will then create a second layer of neurons, this will be our embedding layer which will allow our network to associate similar tiles with each other, e.g. say we have a swamp tile that imparts a movement penalty of 2 and a forest tile that also imparts the same penalty then this second layer will allow the network to treat these tiles similarly. Finally we will have 5 output neuron 4 to represent each cardinal direction and the 5th representing staying in place and we will take the neuron with the greatest value as the networks decision.

Throughout the training process for the networks I will be collecting some specific data that I will be using to assess how proficient the network is at completing the obstacle course. The data I would like to collect includes the amount of time the network was trained for, the number of iterations it went through, the number of moves it takes over each iteration,

# 

# Overview

## Philosophy

Philosophical point #1:

I believe there is a lot of value in researching Neural networks or any type of machine learning for that matter as it will likely have a massive impact on future technologies especially for systems that can be automated.

Philosophical point #2:

I gave my Neural network very limited information about the environment it was in to make the project much more relevant to machine learning as having perfect information means that it is likely an algorithm can be made to take the most optimal actions in any situation in that environment.

Philosophical point #3:

I wanted to make two Neural networks in my project, one with a fully connected hidden layer of Neurons and one without to see how the extra hidden layer affected the results. The reason for this was I was that the hidden layer would help the Neural network recognize patterns but because the Neural networks information was so limited I was not sure to the extent it would be able to recognize any patterns.

## Common Questions

What is the project?

This project is simply an environment to train a Neural network to complete a maze and generate some metrics to evaluate how well it has accomplished this.

Why create this project?

I wanted to make this project because I have a large interest in machine learning and Neural networks and I also believe that these technologies will have a massive impact in the future

What’s different?

What’s different about my project is that I am giving the Neural network very limited information about the world around it so that we can see how well the Neural network fairs in these conditions.

# 

# Define the Application

My application is simply an environment to train a neural network to complete a maze and then generate a number of metrics to evaluate how well it has learnt to complete the maze

# What is the application supposed to do

The application will create a maze for the Neural network to train in.  
To accomplish this it:

* Generates a grid of random tiles to a specified size.
* Creates a controllable character.
* Creates a goal for the character to reach.
* Using the Pygame library it renders the world out to the display.

The application will also create a Neural network that will try and learn to complete the maze. To accomplish this it:

* Creates a memory buffer that the Neural network uses during the learning process
* Using the TensorFlow library it creates a Neural network.

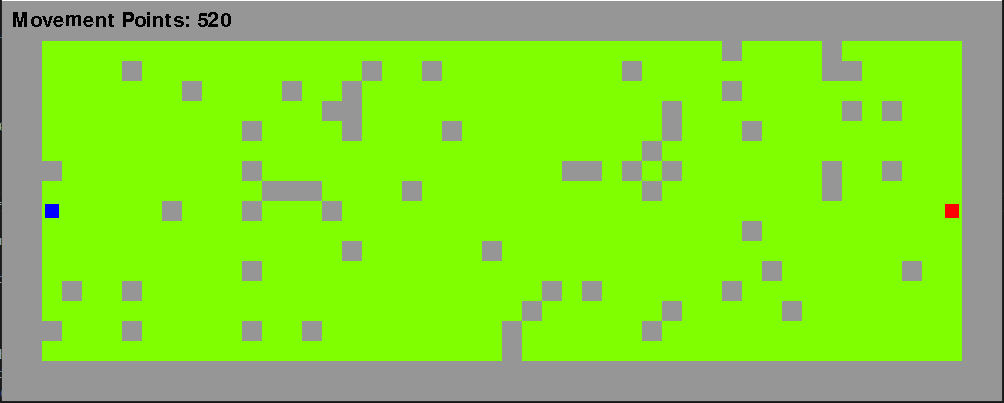
# Experiment

The objective of this experiment is using the neural network I have created to complete a maze, to have this network learn and complete a series of increasingly more difficult mazes and to find out at what point do the mazes become far too complicated for the neural network to complete.

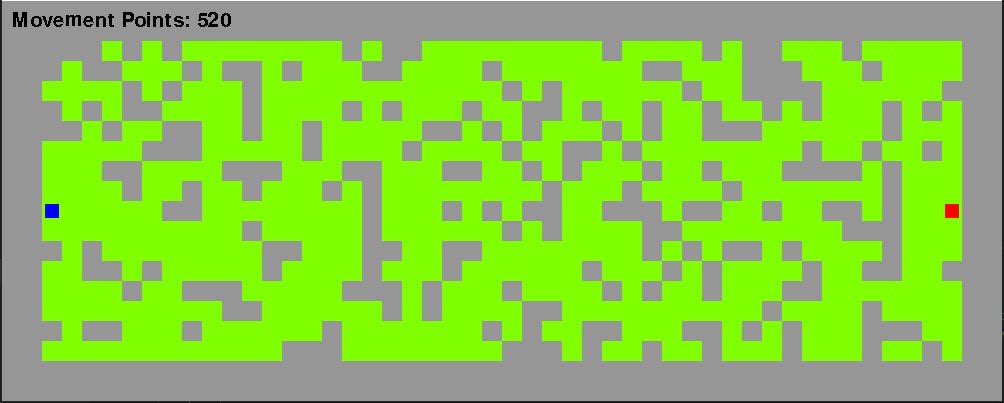
There will be two versions of this network used in this experiment. The first simple version is a network that only has fully connected input and output layers of neurons. The second network will be more complex and will have an input layer, a fully connected hidden layer and a fully connected output layer.

The difficulty level of the maze is the % chance that a wall is placed when the maze is being generated. The % chance of a wall will be the (difficulty level)\*(3), so for example at a difficulty level of 2 then every tile will have a 6% chance of being a wall

**Example of difficulty at level 2:**

****

**Example of difficulty at level 10**



The experiment will start at a difficulty level of 0 and if the neural network successfully learns to complete the maze at that difficulty then we will increase the difficulty by 1 and repeat the process until the neural network can no longer complete the maze

For each difficulty the neural network will first have to learn how to beat it at that difficulty. To do this we will simply generate a maze at that difficulty and have the neural network try to complete the maze and learn from it’s actions, we then repeat this process 1000 times generating a completely new maze each time.

Once the neural network has trained itself at a difficulty level we must now prove whether or not it is able to successfully complete the. To do this we will simply generate a maze and have the neural network try to reach the goal within the move limit and for the sake of accuracy we will do this 100 times generating a new maze each time. This will hopefully give us an accurate idea of how well the Neural network can complete the maze.

From this experiment I intend to capture a number of metrics that we will be able to use to evaluate the success of the Neural networks after they have completed the experiment. The metrics I will be taking will be:

* The average number of moves the neural network took at each difficulty.
* The % rate it succeeded at completing the maze at each difficulty.
* The amount of time it took to complete the experiment.

From my observations prior to this experiment I would like to make some general predictions as to the results of this experiment.

My first prediction is that the more complex network will reach a higher difficulty than the simple network but not by a lot. This is because the decision making for the network isn’t very complex due to its limited range of vision.

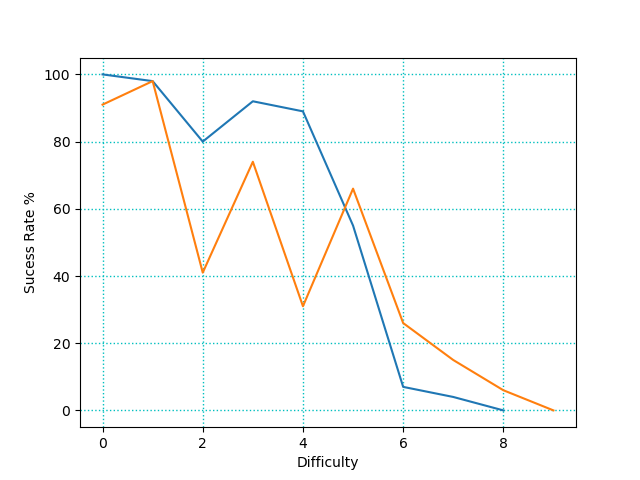
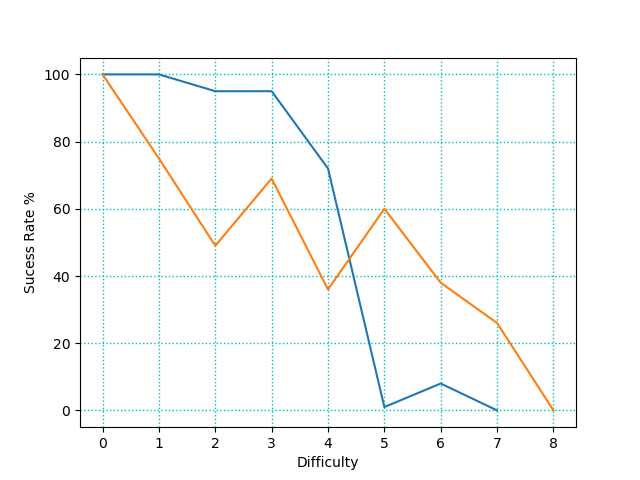
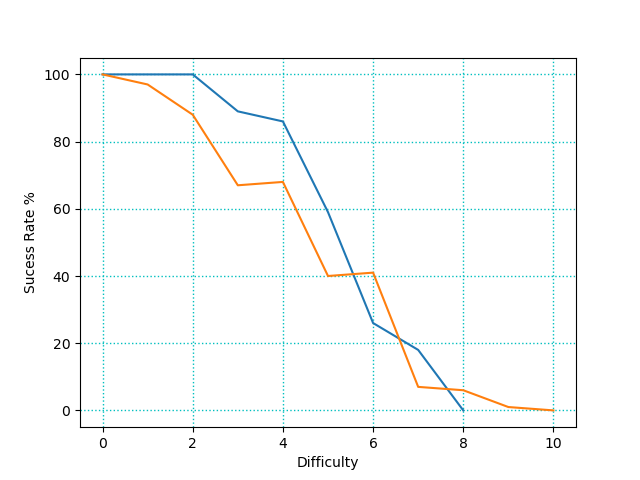
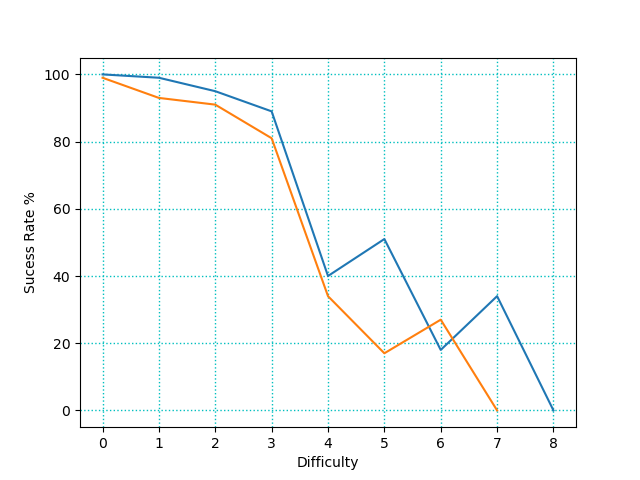
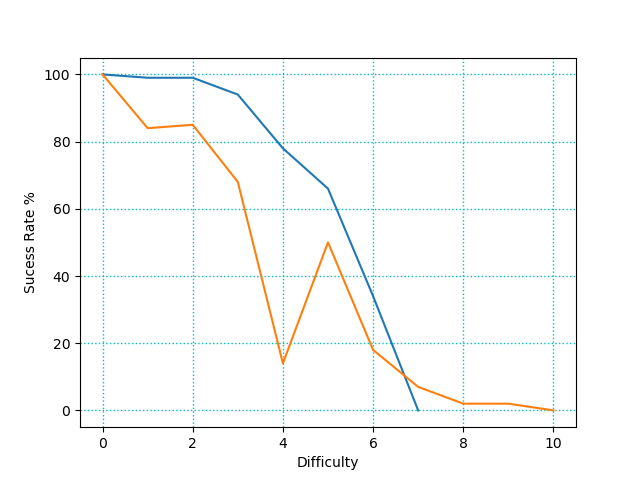
Next and this will seem quite obvious but the number of attempts needed to complete a maze will directly correlate to the difficulty of the maze as will also the % success rate.

# Metrics

I ran the whole experiment 5 times so I could get more accurate results and generated the following metrics from the 5 experiments.

The Blue line on the graph represents the more complex network and the orange line represents the simple network

## Success Rates:



## Moves Taken:

# 

# 

## Time taken to train and complete maze:

# 

# 

# Results:

From the data we generated I’ve made a few interesting observations, while there were many results that I expected there were some very surprising results that I didn’t expect at all.

To start with the completely obvious expected results we can clearly see that as the maze gets more difficult the Neural networks start to struggle and their success rate drops and the amount of moves they take to complete the maze increases. From the success rate graphs we can clearly see that both Neural networks can get to difficulty level 5 (15% chance for each tile to be a wall) without much issue after that however their success drops quite quickly until they can no longer finish the maze.

A result that i did not expect at all was that the simple network would reach higher difficulties than the complex network. From my observations of the networks I believe this is because the more complex network becomes much more cautious and will avoid walls in favor of open spaces but as the difficulty of the maze rises and more and more walls are added this behaviour works against the complex network.

Another strange result I had was that the data I got from the simple network was very sporadic and inconsistent. I speculate that this is due to it learning a bad behaviour and then later getting out of that behaviour.

The time taken to train and complete the maze was incredibly consistent, I got almost the exact same shape and results every time I ran the experiment. What was surprising about this was that the complex network took less time on earlier difficulties and more time on the later difficulties. This I think is due to the complex being more efficient in early difficulties as we can see from the number of turns taken but as the success rate drops for both networks this makes the simple network faster and takes less time since a higher success rate means completing the maze faster and thus less time taken. Since the more complex network has the extra fully connected layer the learning process takes a bit more time as we can clearly see from the very height of the graph where the complex will always reach a much higher time as compared to the simple network.

From observing the network during and after it’s training I made a few interesting observations.

One observation I made was that both networks as they trained would start to favor either the top or bottom half of the maze and they would become much better at solving the maze from their favored half as opposed to their weak half. This reminded me of how people are either right or left handed.

I also observed that the complex network would develop some more complex behaviours, for example it would start to avoid what I call the horseshoe trap which was a group of walls in the shape of a horseshoe that if the network wandered into would struggle to get out of. I also noticed that it would learn to favor open spaces instead of tight corridors

# Is there a precedent for this application?:

There are many many examples of neural networks doing a variety of different tasks from trying to recognize images to recognizing speech to solving board games and just like mine solving mazes.

What makes mine different is that I want to see how the Neural network performs when it’s given a very limited amount of information about its environment.

# 

# Project Milestones

My first major deadline was to do some initial research into my project. I wanted to find what libraries I could use, learning algorithms, etc. I started researching in early October and I gave myself a deadline of mid-October.

After I had completed my research I started working on creating a Neural Network with the OpenNN library and also creating a maze in C++ for that network to solve. I planned on working on this from late October to early December. However, I stopped working on this in late November due to the difficulty I was experiencing working with the OpenNN library.

After the difficulty I faced with OpenNN I decided to switch to TensorFlow and Python and make a very simple test network to see how feasible it was to make the neural network I wanted. I started this in late November and finished making my network early December.

Next I worked on creating a maze for the neural network to train in and rendering it out with the Pygame library. I started this before the Christmas Holidays and finished working on it halfway through the Holidays before my surgery on the 19th of December.

Once I had returned from the Christmas holidays I began working on creating a neural network that could learn how to complete my maze. I knew this would be one of the more difficult parts of my project so I gave myself until the end of February to work on this. I managed to successfully create a neural network that could learn to finish my maze but it was not as good as I wanted it to be so I took a bit more time to improve it until I felt it was competent enough at completing the maze and finished working on this at about mid-March.

The final part of the project I had to work on was to create an experiment to show my findings and provide some insight and metrics to what I found in the project. As this was the final thing I worked on this up until a week before the final deadline of the project.

Throughout the project I felt I made good steady progress over the academic year towards completing each milestone. I did however find it very difficult to make hard deadlines for specific dates due to my complete inexperience with neural networks and how hard it would be to create and train one so I had no context for how much work it would need.  
The one major deadline I set myself when I first started working on the project was for the end of January to have a working Neural network that could learn and solve a maze however due to the difficulties I had working with OpenNN and how hard I was finding it to work with the injuries to my wrist I changed that deadline to the end of February.

# Project Review and Conclusions

Overall I feel that this project was a success. The main part of my project was to create a network that could solve a maze and that’s exactly what I accomplished and I created a nice experiment that generated a lot of data that led to donme interesting results and finding. A lot of these results I did not expect, especially how the more complex behaviour of one network over the other could become a detriment to it.

I did however want to expand on what I did a lot more, there could be a lot more I could add to the experiment such as passing in a Neural network that could see much further into the experiment or properly adding in different types of tiles and rewarding the network appropriately when it moves into them etc. etc. etc. I really feel if I had the time I could add quite a bit more to this project and I felt there were a lot of issues that really slowed down my project. One issue that slowed down my project was that I spent about a month trying to get OpenNN working since I really wanted to work in c++ which didn’t work out very well at all due to OpenNN not having much tutorials or reference material to use to learn how to work with it. This is why I think scrapping everything I had done with OpenNN and C++ and switching over to TenserFlow PyGame and Python was the best decision I could have made for my project as it made developing a Neural network a lot easier and made my project much more successful.

Although the biggest issue that I faced while developing my project were external issues. The first issue was that I fractured my wrist near the start of the academic year and spent collectively about 16 weeks in a cast which massively impacted how efficiently I could work. After I had recovered I then had to completely change my working environment due to the Covid-19 lockdown.

Since I had never worked with Neural networks and TensorFlow before I learned quite a lot from this project and if I were to start this all over I think I would like to increase the scope of the project and include a competing Neural network to the maze with it’s own objective that would interfere with the original Neural network. I would also like to put a lot more work into how I reward my Neural network as I feel I could expand on what I currently have

# 

# References

# 

|  |  |  |
| --- | --- | --- |
| **Referenced Publication** | **Citation** | **Reference** |
| Report | Mastering the game of Go with deep neural networks and tree search | *Silver*, D. (2016). Mastering the game of Go with deep neural networks and tree search. nature |
| Website | (Violante Andre 2019) | Violante , A. (2019, March 19). Simple Reinforcement Learning: Q-learning [Online]. (URL <https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56>). (Accessed may 2020). |