1.0 Introduction:

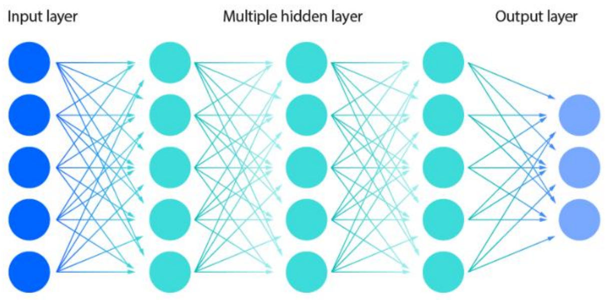
For this folder assignment I have decided to make a project that teaches “cars” to drive on a handmade track, using “*Neural Network*”.

For this project I’ve used resources from a Youtube channel named TheCodingTrain. Here I’ve gone through 10.1 to 10.18 of his lectures, which cover the basics of building a Neural Network from scratch, and the implementation of a Matrix library used in calculations for the Neural Network. He uses the code language JavaScript in his lectures, so I’ve had to rework the code for C# and make sure it was compatible within the Unity Engine.

Through research and “taking classes” at The coding Train, I have learned how a Neural Network works and how to take the knowledge a step further, implementing the network for use in this project. My focus for this project has been the implementation of the programing logic and my goal has been to make carObjects learn a route on a track by themselves, from scratch. I have therefore not focused on making an app for others to use, so the assets, scene and UI doesn’t contain more than what’s necessary to achieve my goals.

2.0 Neural Networks:

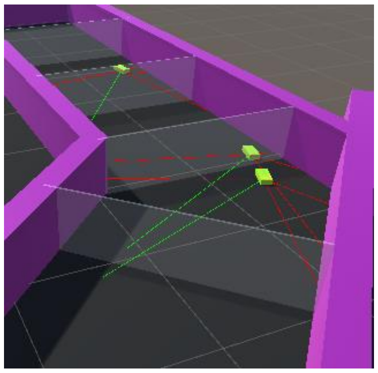
Neural Networks are connections of inputs made by simulating how neurons are working in our brains. The goal is to give something artificial the ability to “learn” different operations. Examples of this are *pattern recognition* (ex. numbers), simulation of *processes in the human body* (ex. stress), and *vehicle movement*. In short: everything that has the possibility of learning from trial and error.



A Neural Network takes in a set of inputs and gets a set of outputs after traversing through a set of hidden layers. The number of inputs, outputs, and layers (and their sizes) are decided based on the task at hand. In-between each transition from one node to another, a weighed sum is added to the number that gets transported. This is to give some variation to the outcome, so that there will be a slight difference for the objects inheriting from this network (in the case of testing hundreds of objects at the same time). When we have an output, instead of inputting this number back into the network as an input, we perform something called a “*Back Propagation*”, meaning that we traverse through the network backwards, looking for errors and adjust the weights to get as small errors as possible. When the numbers reach the start of the network, we have a much more suitable number to work with, inserting in the object, and going forward through the network again.

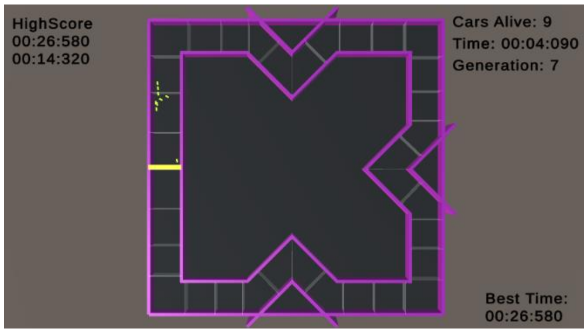
3.0 Learning cars to drive:

I have implemented this Neural Network and have used it to teach cards to drive on a track. To be able to use the Neural Network, a set of rules and physics needed to be implemented in the carObject. The goal of the Neural Network has been to manipulate the card's steering and velocity so that it follows my set of rules as accurately as possible, after many trails and errors.



Firstly, to get a direction, a car has five raycasts which search for either a wall or a clear passage. As you can see from the picture, the raycasts turn green when there aren't any obstacles in the way.

The inputs from the raycasts are used to feed the Neural Network in a way so that the output shows the car how much in either direction it needs to steer, combined with its calculated velocity. When the car receives an output, it doesn't steer all the way at once. Since this is performed each tick, steering all the way will lead to overshooting. So, the result is a car that drives on the track as a player would with the use of the "WASD"-buttons.



I've added a simple UI to display what I think is the most important data regarding looking for an improvement in the cars. For the cars to be able to finish the whole track, over 150 iterations need to be performed. I will not be able to show this in my review video, but I hope to display some improvement.

As for the camera, I've kept it simple, as the player sees the track in 2D view. Even if the scene is built in 3D, I've done it this way to be able to have the whole track in sight throughout the simulation.

4.0 Future thoughts and implementation:

As for now I believe that I have checked off most of the boxes regarding this assignment. If I were to expand this project in the future, the first thing I would implement is a file saving system so that my cars don't have to start from the beginning of the learning process each time I run the project. I have spent a lot of time implementing file systems in both VisSim and the bachelor project this semester, so to spare time for other assignments I'm not adding it to this project.

I would also like to make a playable game, based on heavily trained cars, to make a challenge for any player. I think I also would have improved the scenery a bit, implementing the physics code from VisSim to make the cars able to drive on elevated meshes.

5.0 Conclusion:

This field has opened my eyes to a whole new world of coding, giving me a glimpse of which opportunities lay in the power of coding. I have worked hard trying to grasp the fundamentals of Neural Networks and have come a long way with managing to implement it into my own project. I am glad that I've achieved what I worked towards; making "cars" able to teach themselves how to drive on a track.

6.0 Resources:

- The Coding Train (2017), *Neural Networks – The Nature of Code*. Url: https://www.youtube.com/watch?v=XJ7HLz9VYz0&list=PLRqwX-V7Uu6aCibgK1PTWWu9by6XFdCfh&ab_channel=TheCodingTrain