# Deep Learning Project

# Self-Driving Car

## Team 5


| | |
|---|---|
| **Khaled Medhat Mahmoud** | **18P3557** |
| **Mohamed Magdy Mostafa** | **18P5160** |

# I. Introduction

In this project we were instructed to train a neural network machine learning model to simulate a self-driving car in the udacity car sim made using Unity.

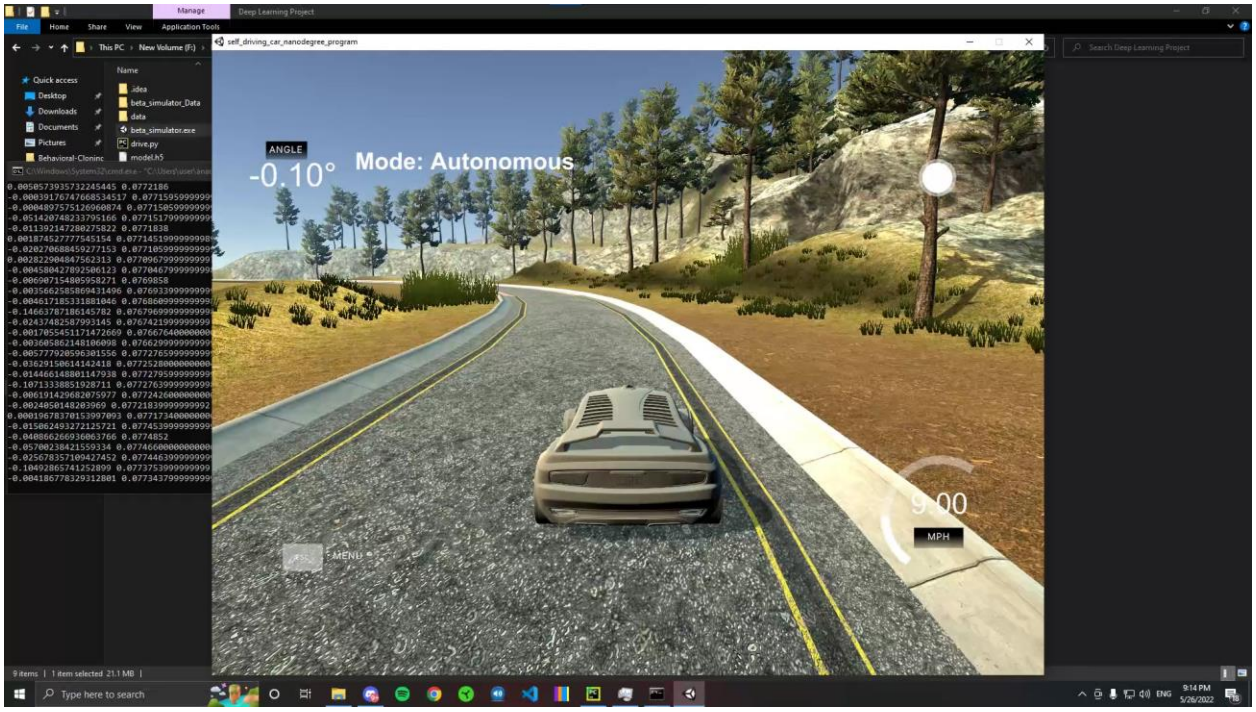
We've learned a lot about Tensorflow, Keras, NumPy, and general machine learning concepts that were a big challenge and a fun learning experience.

# II. Failed Attempts

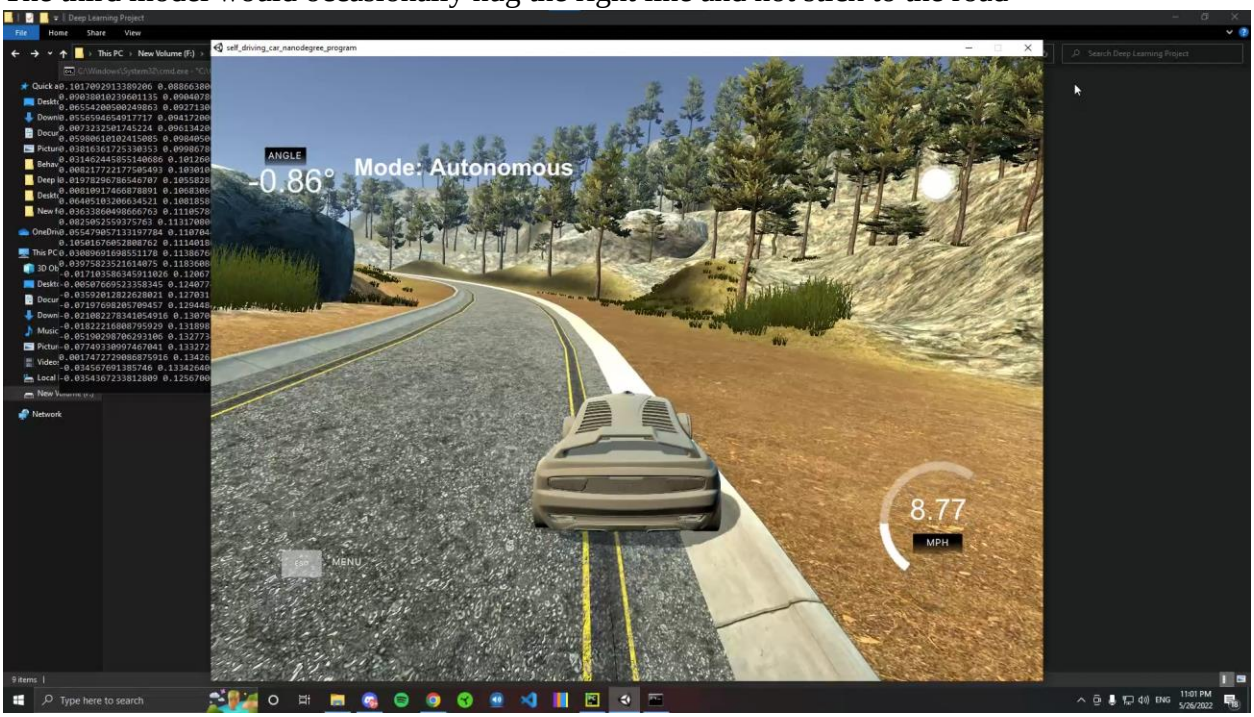
We have gone through over 20+ models over the past week to try to perfect it. The first couple of models were mostly made of a lambda layer and one or two convolution layers followed by one MaxPooling and lastly, a dense layer.

Sadly, we didn't record all the failed attempts in video but took some screenshots and small clips, some of them are:

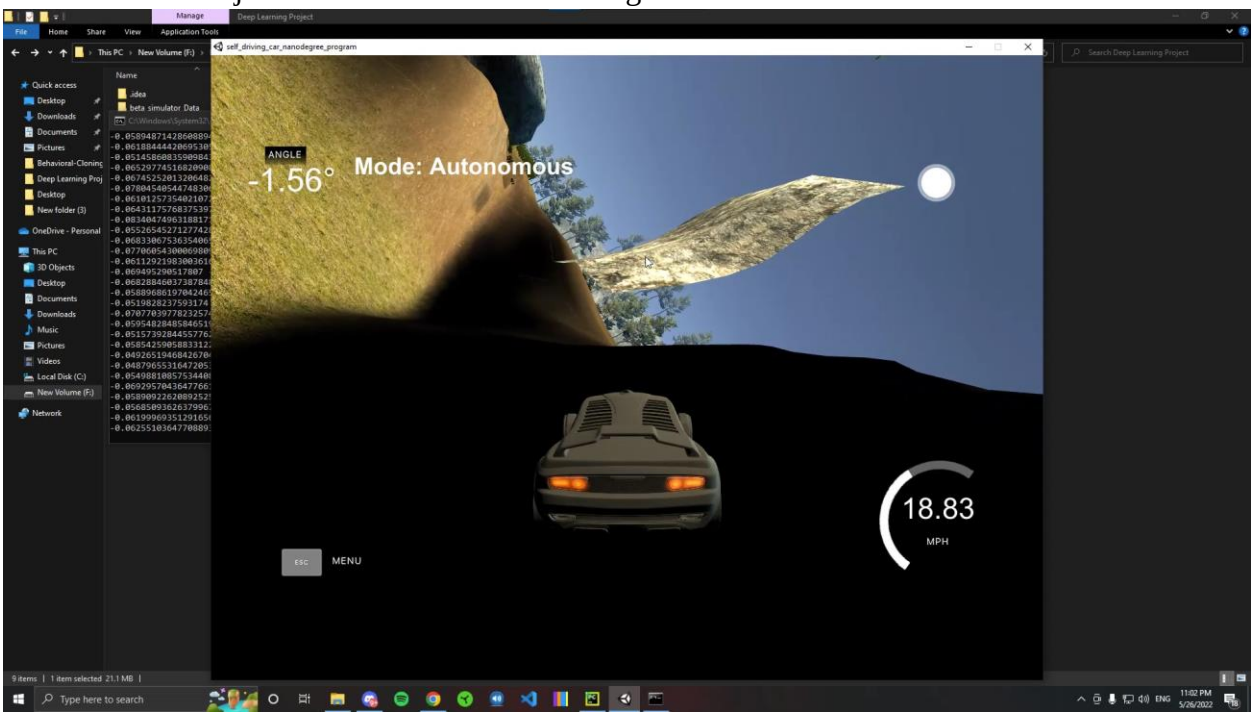
1- Our very first model would go in circles as it always had its angle set to 25 degrees.
2- The model was slanted to the right side and went off the side of the road often.



3- The third model would occasionally hug the right line and not stick to the road




4- The fourth model just went off the road and straight into the water



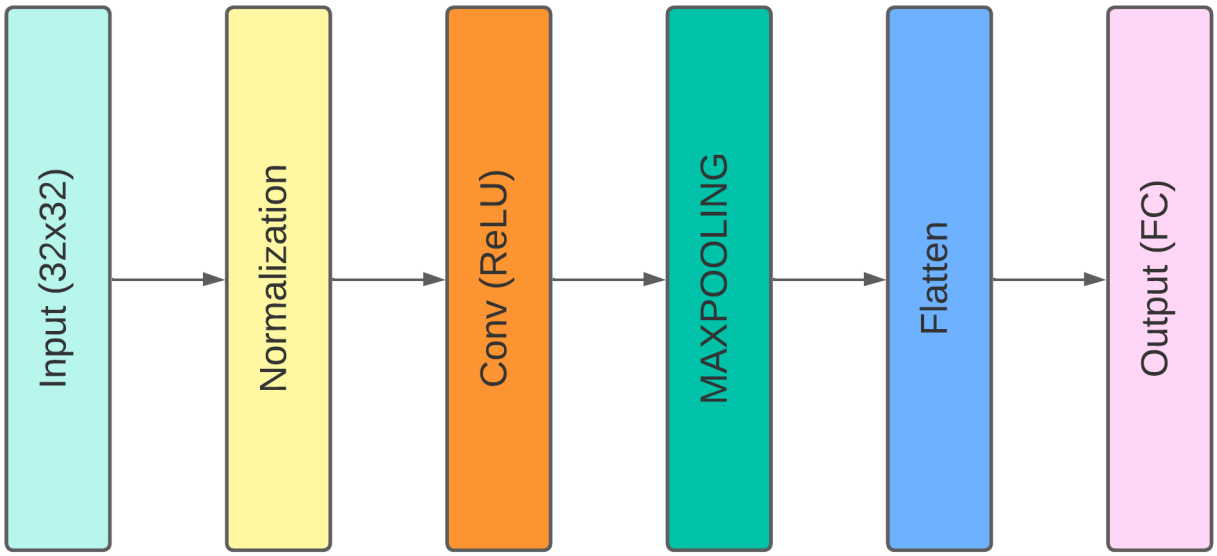
## III. Training the Model and Data Augmentation

As something was very clearly off in the training model, we kept trying for hours more combinations of conv/pooling/dense layers, and tried multiple activations such as ReLU, ELU, etc, but none of these combinations worked.

Then to increase the amount of data we have, instead of just running the track multiple times, we decided to implement some data augmentation methods, so we added the original images with the measurements, then we added the flipped (LR) images of the originals with -ve measurements to ensure accuracy.

Then we added the right and left camera angle and since they are a bit off centered, we added an offset of -0.3 and +0.3 to the right and left camera measurements respectively.

Then we reduced the model layers to just one lambda layer for normalization purposes, followed by a conv layer with filter size 15, a kernel size of 3, strides equal to 2, and a ReLU activation function, then that layer is followed by a dropout of a (0.4) chance (which was selected by trial and error), followed by a MaxPooling layer, flatten and then dense layers.




Then we thought that the model might be getting affected by the terrain in the back, so we cropped the top side of the photo which only includes the terrain in the background and not the actual road.

This model actually worked perfectly but would every now and then do sudden movements so we thought that the model might be getting influenced by the terrain in the background, then we cropped the photo to only show the middle part with the road without the car chassis and the trees, and we resized the picture to 32x32 for speed purposes (we would've returned the image to the original size of 140x80 if the resizing affected the model negatively but it didn't so we stuck with it).

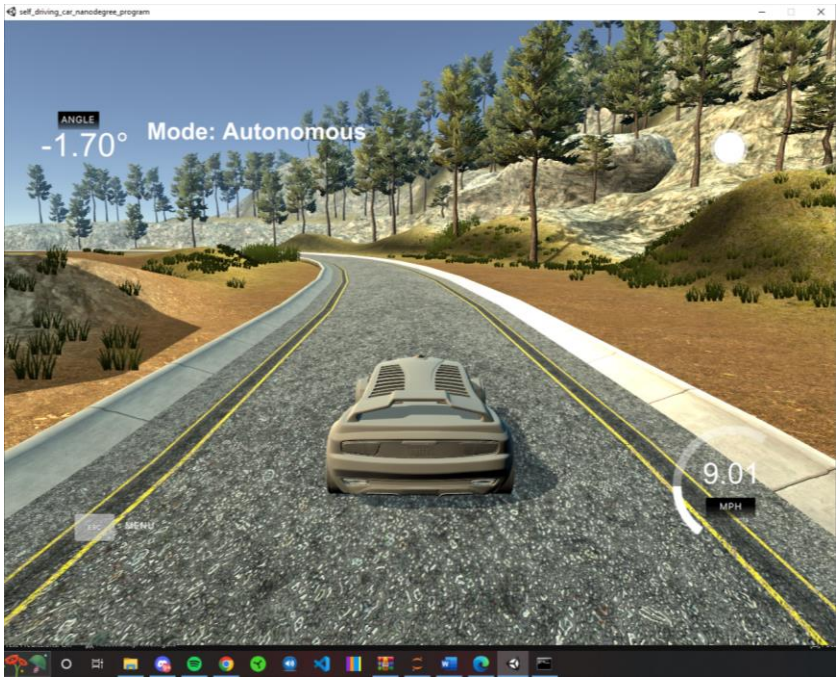



Photo of the model after training.

Since we're only using 140x80 of the images themselves, we've altered the **drive.py** file accordingly to resize the image_array to the appropriate size and we've included the following files in the .zip:

- drive.py
- model.py
- model1.h5 (model to run the first track)
- model2.h5 (model to run the second track)
- run1.mp4 (video footage of the autonomous track 1 run)
- run2.mp4 (video footage of the autonomous track 2 run)