ANKARA UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

COM 3551-B PROJECT REPORT

Self-Driving Car in a Simulation

Batuhan Salmanoğlu

17290571

January 2021

DEMO: https://www.youtube.com/watch?v=ciLmBspm66k&t=244s

TABLE OF CONTENTS

Abstract:

In this report, I explain what I did in com3551-B lecture project. Project is self-driving car in a simulation. Also, I write some information about self-driving and existing self-driving project (Nvidia).

1. Introduction:

1.1. Problem:

One of the important problems of today is traffic accidents. Every day, every hour someone dies for this reason. Otherwise, most of the people stuck in traffic, every day. Also, the physical difficulties of driving can be included in problem. As a result, human error in daily traffic is main problem in project.

1.2. Project's Aim:

In traffic, eliminating human error so prevent traffic accidents and driving difficulties. To do this use self-driving car that is creating by artificial neural networks. It is important to solving the problem. Additionally, testing artificial neural network in simulation and seeing the results and improving the solution is aim of the project.

Self-driving car that are an important part of today. My project is about this. In the project, firstly I collected data from simulation (manual driving) and trained this data to CNN (Convolutional Neural Network) then test it in simulation.

2. About Self-Driving:

2.1. Self-Driving Cars Definition:

Self-driving vehicles are cars or trucks in which human drivers are never required to take control to safely operate the vehicle. Also known as autonomous or “driverless” cars, they combine sensors and software to control, navigate, and drive the vehicle.

2.2. History of Self-Driving cars:

In GM’s 1939 exhibit, Norman Bel Geddes created the first self-driving car, which was an electric vehicle guided by radio-controlled electromagnetic fields generated with magnetized metal spikes embedded in the roadway. By 1958, General Motors had made this concept a reality. The car’s front end was embedded with sensors called pick-up coils that could detect the current flowing through a wire embedded in the road. The current could be manipulated to tell the vehicle to move the steering wheel left or right.

In 1977, the Japanese improved upon this idea, using a camera system that relayed data to a computer to process images of the road. However, this vehicle could only travel at speeds below 20 mph. Improvement came from the Germans a decade later in the form of the VaMoRs, a vehicle outfitted with cameras that could drive itself safely at 56 mph. As technology improved, so did self-driving vehicles’ ability to detect and react to their environment.

2.3. Nvidia Self-Driving:

NVIDIA DRIVE™ solutions span autonomous vehicle development from the cloud to the car, helping manufacturers collect data, train deep neural networks, and test, validate, and operate self-driving cars.

3.Architecture of Project:

All my study in this section has been done on the colab platform.

3.1. Data:

Data in this project is image. I use approximately 30.000 image and all images crated by myself. There are three types of images; left-camera, center-camera, right-camera.

Example left-camera, center-camera, right-camera images from my dataset:

center-camera

right-camera

left-camera

As you can see above, there are images of every moment of driving from 3 different angles. This driving made by me and recorded by simulation. At the same time, the steering angle of the car is recorded for each image.

, C:\Users\batuhan\Desktop\AI PROJECT\IMG\right_2020_12_12_20_59_43_025.jpg, -0.1638867

In this picture steering angle = -0.1638867

I use NVIDIA drive CNN model in this project. Nvidia model's aim is in each frame predict best and safety steering angle to drive. So, I prepare dataset and steering angle most suitable format the NVIDIA model.

3.2. Prepare Dataset to Train:

Datasets need to some preprocess before training CNN. These preprocesses include:

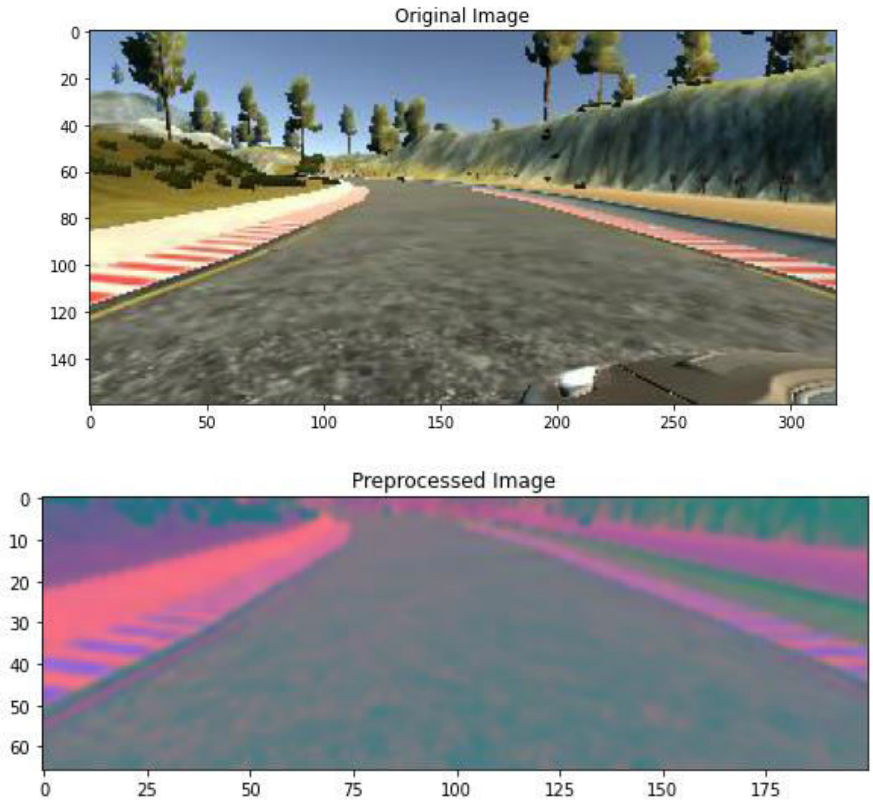
- Resize image: To make smaller and reducing pixels.

- Apply Gaussian blur: Also known as Gaussian smoothing, is used reduce image noise and reduce detail also provides image structures of different scales.
- RGB to YUV colors: YUV is used because of more suitable for NVİDİA model and YUV is better for machine vision implementations than RGB due to the perceptual similarities to the human vision.

To do all of these using OPENCV is enough. There is related code below.

```
def img_preprocess(img):
    img = img[60:135,:,:]
    img = cv2.cvtColor(img, cv2.COLOR_RGB2YUV)
    img = cv2.GaussianBlur(img,  (3, 3), 0)
    img = cv2.resize(img, (200, 66))
    img = img/255
    return img
```

These are all for prepare images and below is the initial and final version.



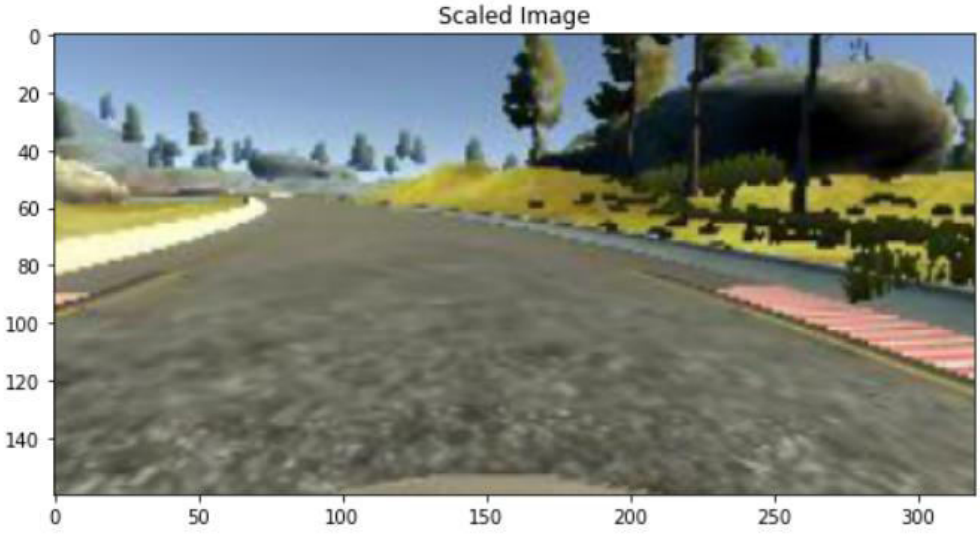
3.3. Data Augmentation:

I use data augmentation because without augmentation CNN model did not succeed in learning. After augmentation both data variation and succeed in learning increase. For this, four types of augmentation were enough, and I was use imgaug and OpenCV library. I used imgaug for scale, different brightness and translate. For flip I used OpenCV.

1. Scale:

Create image with 1 and 1.3 ratio scaling.

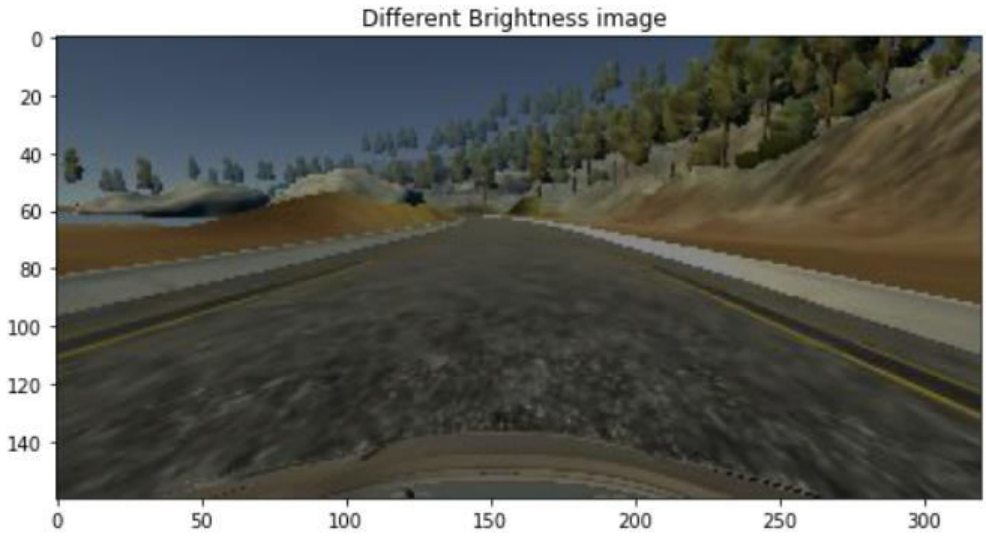
```
def scale(image):
  scale = iaa.Affine(scale=(1, 1.3))
  image = scale.augment_image(image)
  return image
```



2. Different Brightness:

Change brightness to random brightness.

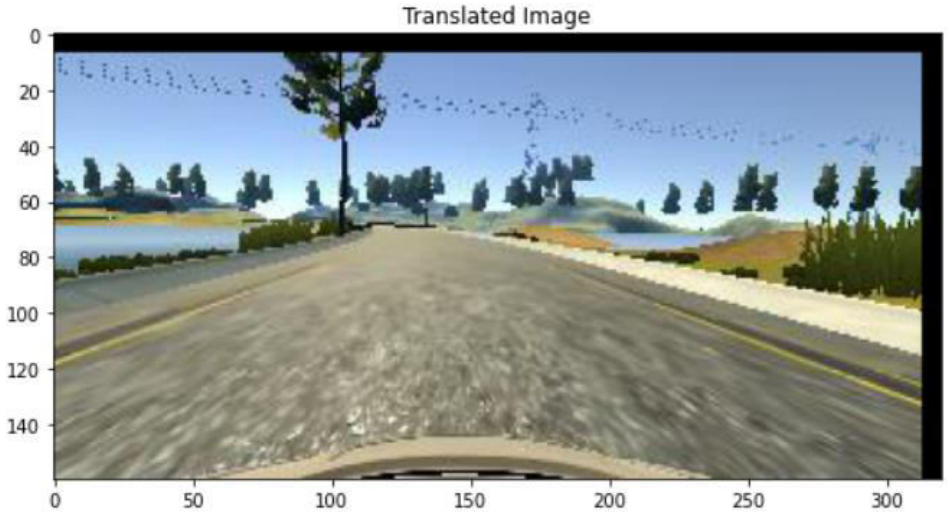
```
def different_brightness(image):
    brightness = iaa.Multiply((0.2, 1.2))
    image = brightness.augment_image(image)
    return image
```



3. Translate:

Translate image on x and y axis.

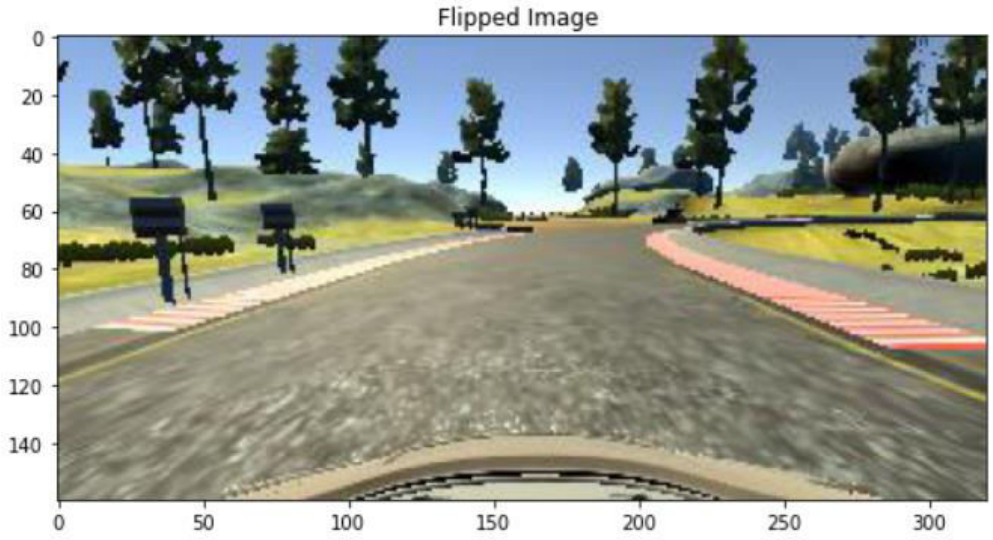
```
def Translate(image):
  Translate = iaa.Affine(translate_percent= {"x" : (-0.1, 0.1), "y": (-0.1, 0.1)})
  image = Translate.augment_image(image)
  return image
```



4. Flip:

Create image generated by a mirror-reversal of original across.

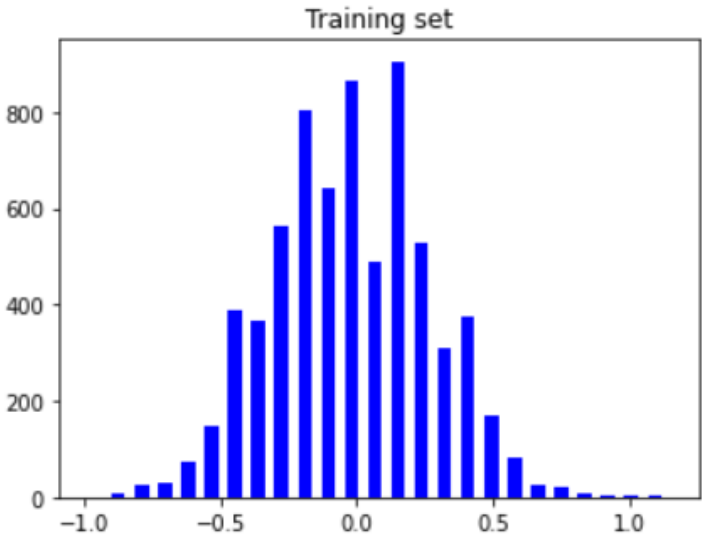
```
def flip(image, steering_angle):
    image = cv2.flip(image,1)
    steering_angle = -steering_angle
    return image, steering_angle
```



3.4. Training – Validation Set:

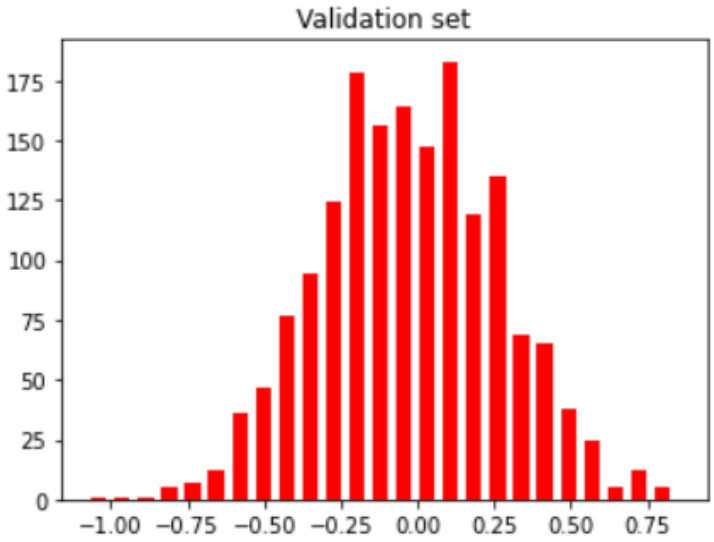
Training data set which set of example data to train CNN model. I used approximately 7000 example parameters in project. Validation data set which is set of example data to check already trained CNN model and in this project Validation set consisting of approximately 1500 sample parameters.

```
Training Samples: 6820
Valid Samples: 1706
```



y-axis: number of images

x-axis: steering angle of image

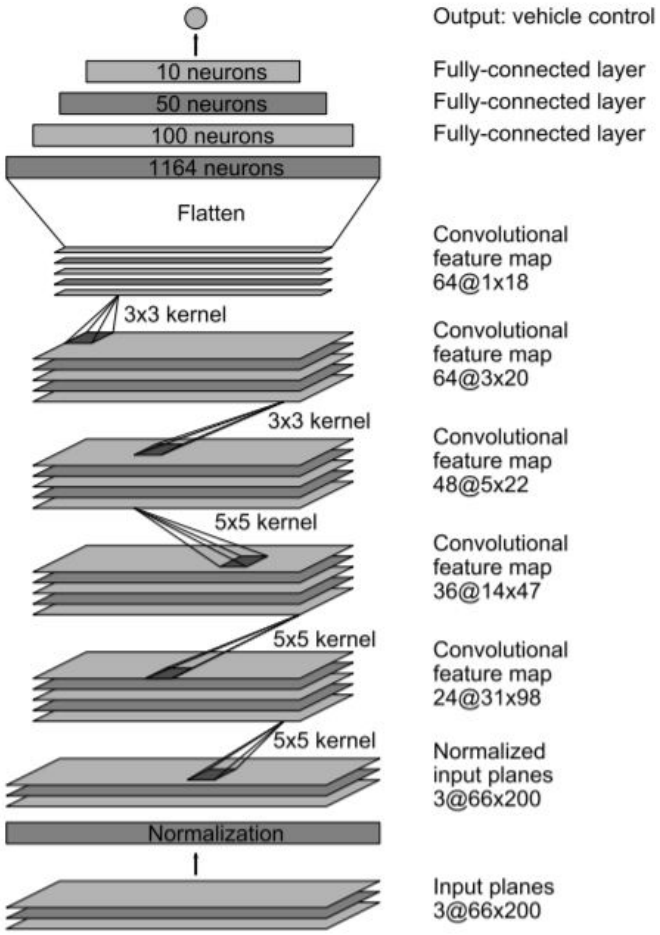


y-axis: number of images

x-axis: steering angle of image

3.5. CNN Model:

I used official NVIDIA self-driving cars convolutional neural network design. The design official I shared below is from the Nvidia documentation.



Like the above design, in my project there are five convolutinonal2D and four Dense keras layers. For all these layers activation function is elu which is very similar to relu function and learn rate in this model 0.001. It is very common rate for this type of models.

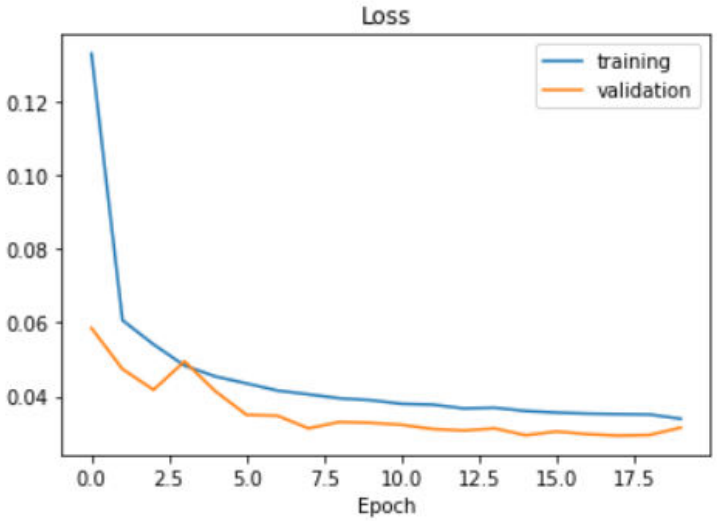
Models Summary

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 31, 98, 24)        1824
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 47, 36)        21636
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 5, 22, 48)         43248
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 20, 64)         27712
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 1, 18, 64)         36928
_________________________________________________________________
flatten (Flatten)            (None, 1152)              0
_________________________________________________________________
dense (Dense)                (None, 100)               115300
_________________________________________________________________
dense_1 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_2 (Dense)              (None, 10)                510
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 11
=================================================================
Total params: 252,219
Trainable params: 252,219
Non-trainable params: 0
```

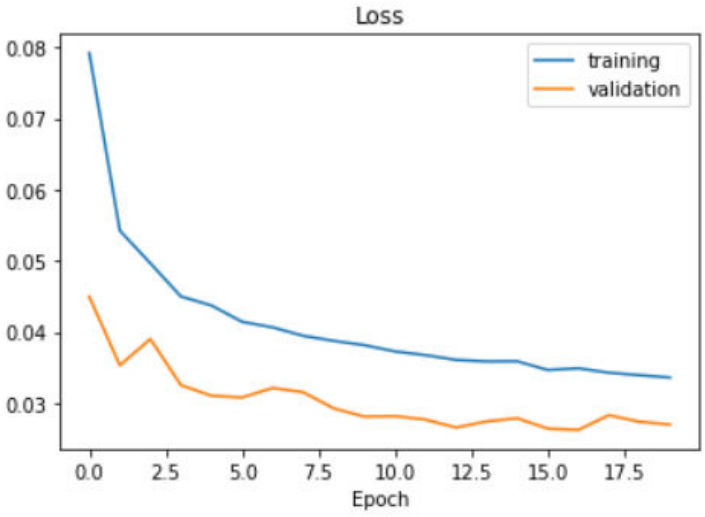
3.6. Model Results:

I am traying to change some parameters and to see models result. So get best model to self-driving. These parameters are epoch, steps-per-epoch and learn rate. I made sure that there is no overfitting here. Also, I got enough accuracy to self-driving. I share a few results below.

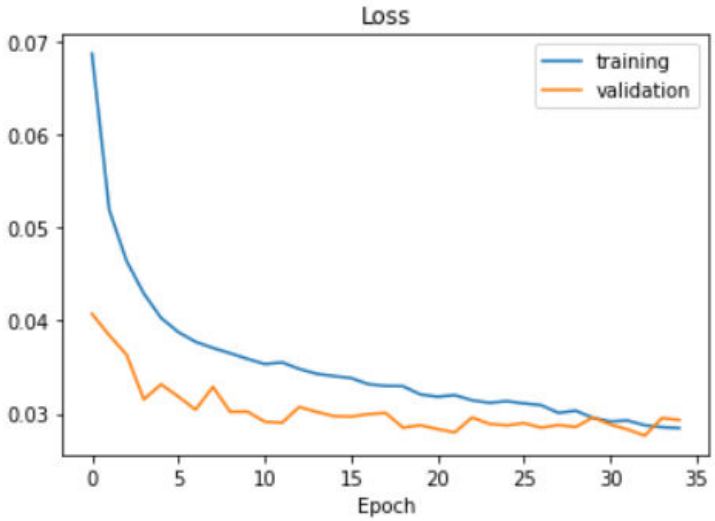
1. Epoch: 20
   Steps per epoch: 300
   Learn rate: 0.001



2. Epoch: 20
   Steps per epoch: 300
   Learn rate: 0.0005



3. Epoch: 35
   Steps per epoch: 400
   Learn rate: 0.0005



4.Test and Result:

In the previous section, I trained the model in colab, I download it from there. Then I created python script, load model there. Communication with simulation and script provided by server-socket method. Simulation send instantly right-center-left images to script then model predict steering angle and return simulation this prediction. Additionally, all of this happened in Anaconda environment.

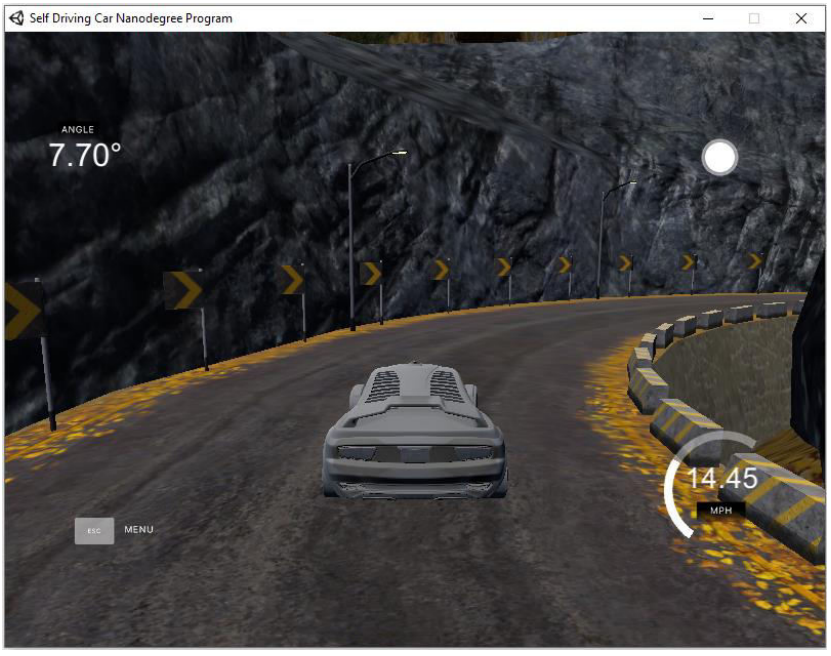
Result:

In simulation there are two tracks. First one is where I create the data, so all images related this track. Consequently, auto-drive in this track works smoothly and completed successfully.

Track 1 from auto-drive.

Second Track which is no images in the dataset. Auto-drive although never seen this track before completed approximately 90 percent success.

Track 2 from auto-drive.



5. How It Is Improve:

Working in the more complicated simulation, so more parameters for instance; more complex sensors (lidar, gps etc.), more realistic real-world conditions. With more parameters we may design more powerful neural network. Last of all, in project, neural network only predicts steering angle with camera input, this is improving with more parameters.

References:

- Keras library, URL: https://keras.io/api/layers
- Sklearn Train Test Split, URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
- Imgaug library, URL: https://imgaug.readthedocs.io/en/latest/source/api_augmenters_geometric.html
- Gaussian blur, URL: https://en.wikipedia.org/wiki/Gaussian_blur
- YUV colors, URL:https://www.researchgate.net/publication/281822371_YUV_vs_RGB_-_Choosing_a_Color_Space_for_Human-Machine_Interaction
- Deep learning Course on Udemy, URL: https://www.udemy.com/course/applied-deep-learningtm-the-complete-self-driving-car-course
- Nvidia self-drive project, URL: https://developer.nvidia.com/drive
- Self-driving definition, URL: https://www.ucsusa.org/resources/self-driving-cars-101
- History of self-driving cars, URL: https://www.titlemax.com/resources/history-of-the-autonomous-car/