ANKARA UNIVERSITY

ENGINEERING FACULTY

COMPUTER ENGINEERING DEPARTMENT

RESEARCH TECHNIQUES

COM4061

SELF-DRIVING CAR SIMULATION

PROJECT REPORT

Muratcan Akbay

16290074

Ahmet Sefa Emre

17290099

GAZİ ERKAN BOSTANCI

Aralık 2021

ABSTRACT

Self-driving cars have become a trending topic with the technological developments in recent years. The goal of the project is to train a neural network to drive an autonomous car on the tracks of Udacity's Car Simulator environment. Udacity released the simulator as open source software, and people were in a race to teach a car how to control itself with just camera footage and deep learning. Autonomous driving requires learning to control steering angle, throttle and brakes. The behavioral cloning technique is used to mimic human driving behavior in training mode. This means that the car driven by the user in training mode generates a dataset in the simulator, and deep neural networks model then drive the car in autonomous mode. The three architectures were compared based on their performance.

Although the models performed well for the track they were trained on, the main challenge was to generalize this behavior over a second track in the simulator. Track_1 dataset, which is simple with road conditions suitable for driving, was used as a training set to drive the car autonomously in Track_2, which consists of sharp turns, obstacles, heights and shadows.To overcome this problem, image processing and different magnification techniques were used, allowing us to extract as much information and features from the data as possible.

As a result, the aim of the project is to drive autonomously on Track_2, the new map of the model, after model training and modeling with the data collected in Track_1.

TABLE OF CONTENTS

1. INTRODUCTION

Technology is developing day by day and it creates ways to satisfy people needs. One of the most important need in world is transport. Transport means basically using a vehicle or system of vehicles for getting from one place to another. Purpose of technological advancement is making transportation safer and faster. Autonomous car is a huge development of this purpose. Autonomous cars will be able to eliminate sensing and perception errors, or crashes that result in the driver's distraction, and autonomous technologies won't be subject to the influence of drugs or alcohol. So, that takes incapacitation errors out. From the sample, that accounts for 34% of crashes.

Historical events helped shape modern semiautonomous vehicles. The first step towards autonomous cars was the radio controlled car, called Linriccan Wonder. It was demonstrated by Houdina Radio Control in New York City.It was basically a 1926 Chandler that had transmitting antennae on its rear compartment and was operated by another car that sent out radio impulses while following it. It was one of the most primitive forms of autonomous vehicles.[1]

In last years significant progress in Machine Learning (ML) techniques like Deep Neural Networks (DNNs) has enabled the development of autonomous cars. Several major car manufacturers including Tesla, GM, Ford, BMW, and Waymo/Google are building and actively testing these cars. Recent results show that autonomous cars have become very efficient in practice and already driven millions of miles without any human intervention [2, 3].

1.1. <u>Aim</u>

Our aim in this project is to develop a self-driving vehicle in a simulation environment. In this environment, the car should be able to adjust the gas-brake balance on its own, and increase or decrease its speed where necessary. In this chapter, we evaluate this problem and analyze appropriate solution techniques.

1.2. <u>Problem Definition</u>

Udacity is an open source simulation that environment developed for autonomous car research. In this simulation environment, the aim is to enable a human driver to perform the same behaviors autonomously in his car by deep neural network[4]. this concept is called Behavioral Cloning.

There are 2 different maps in this simulation environment. In addition, there are two different modes in this simulation environment, training mode and autonomous mode. In the training mode, the movements of the car driven by a human user are recorded and this data is used as a dataset. The problem is, after training the deep learning model with this user data, how to use it in autonomous mode and what will be the results of the action that will be given to this trained model on the other map.

1.3 <u>What is Deep Learning</u>

Deep learning is a “machine learning” method consisting of multiple layers that predicts results with a given data set. Deep learning, machine learning and artificial intelligence are terms that have different meanings. Deep learning, machine learning; machine learning can be summarized as a sub-branch of artificial intelligence.

Deep learning can be conducted supervised, semi-supervised or unsupervised. In deep learning, it learns distinctive features by itself with a large number of data entries.

The more data input for the learning process, the more successful it will be. Data passes through multiple layers. The upper layers are those that bring out more detail.

2. METHODOLOGY

2.1. Pyhton

Studying with AI algorithms most of the time hard and takes time so it is a need to study with environment which makes testing and readibility easy and mostly focus on developing on the other hand, structure of the environment also should be appropriate.

Python is a powerful language to study AI and other areas because it has a simple and readabile syntax which makes things easier to understand and use effectively also the frameworks and libraries of python can be used to solve many problems in diffrent areas.So,it is becoming more popular among developers.

2.2. Numpy

Studying on artificial intelligence requires a lot of complex numerical and mathematical operations.To do this effectively and easily we use open-source python library which is Numerical Python or shortly NumPy.It makes these operations simple and effective.

2.3. CNN Architecture

CNN is mostly used for analyzing visual imagery and it is a artificial neurol network. Convolutional neurol network also known as shift invariant.It is different than other artifical neurol network classes by including convolution layer.

Convolution layer has some neurons connected with next layer. These neurons has many connections and same weight sharing by these connections. Connections come together and create sets and sets together create a convolution kernel. Convolution kernel apply the same convolution operation on the outputs of a set of neurons in the previous layer.

There are two significant benefits of using convolution layers. First one is, convolution layer decrease the number of trainable weights by allowing sharing of weights among multiple connections and thus significantly reducing the training time. Other benefit is convolution kernel application is a natural fit for image recognition same as the human visual system which extracts a layer-wise representation of visual input.

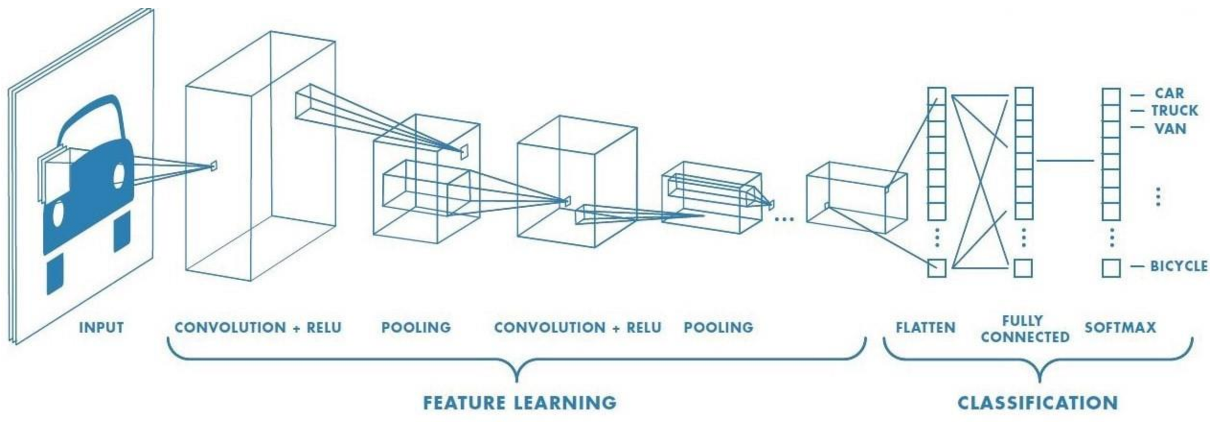


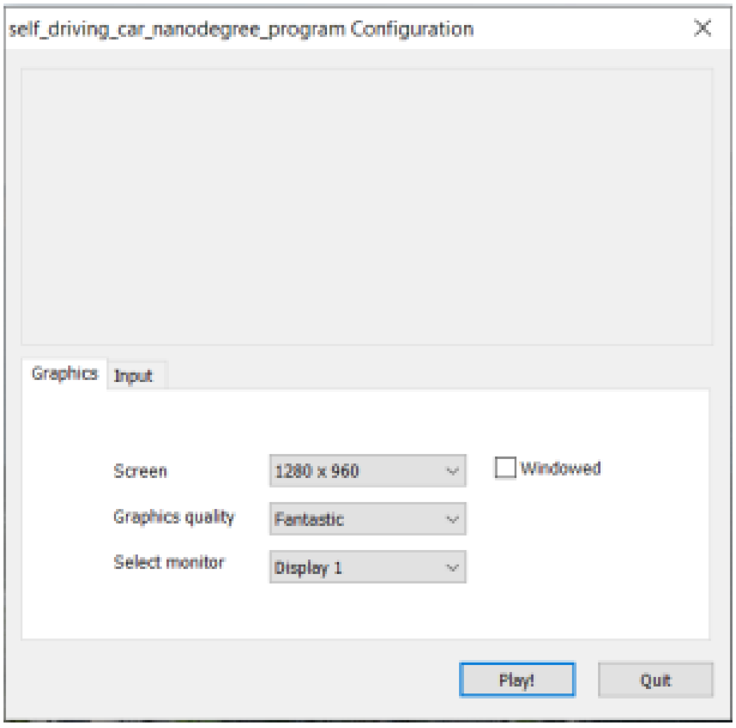
Figure 2.1 : Architecture of CNN

2.4 <u>Keras</u>

Keras is an open-source software library that makes a Python interface for artificial neural networks. Keras behave like interface for the TensorFlow library. Languages with a high level of abstraction and inbuilt features are not enough quick and building custom features in then can be hard. But Keras runs on top of TensorFlow and is relatively fast. Keras is also deeply integrated with TensorFlow, so we can create easily customized workflows

3. UDACITY SIMULATOR AND DATASET

Udacity has developed an open source simulator environment to research and project about self-driving cars. It is built on Unity, the video game development platform. The simulator has a user-friendly interface in terms of resolution, control and graphics settings.


self_driving_car_nanodegree_program Configuration

Screen: 1280 x 960 ☐ Windowed
Graphics quality: Fantastic
Select monitor: Display 1

Figure 5.1. Configuration screen

Graphics and input settings can be changed according to the user's preference or the performance of the device used. When the user presses the play button, the simulator interface screen opens.

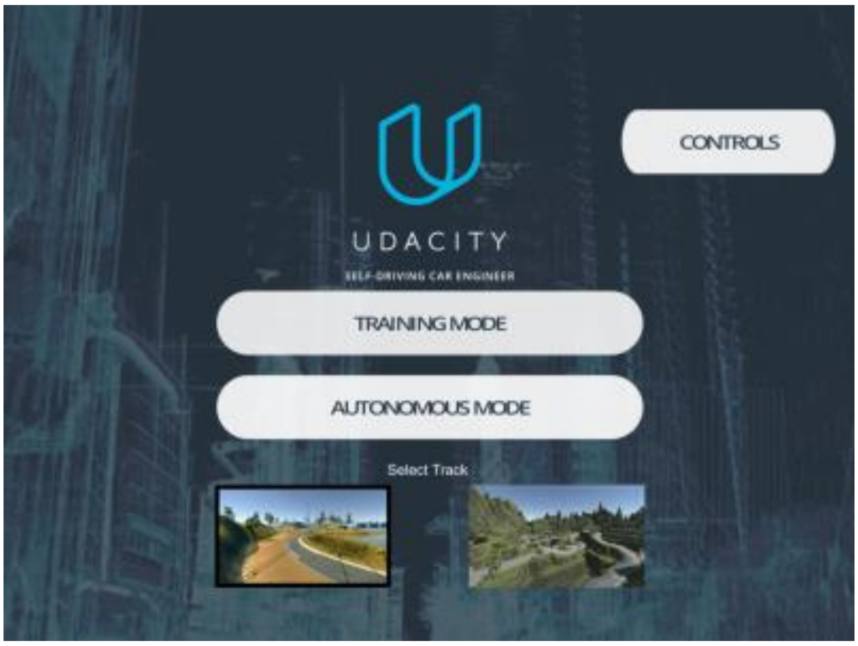


Figure 5.2. Main menu of the simulator

If the user clicks on the "Controls" button, user will switch to the part that shows how to move the car and how to record in the simulation.

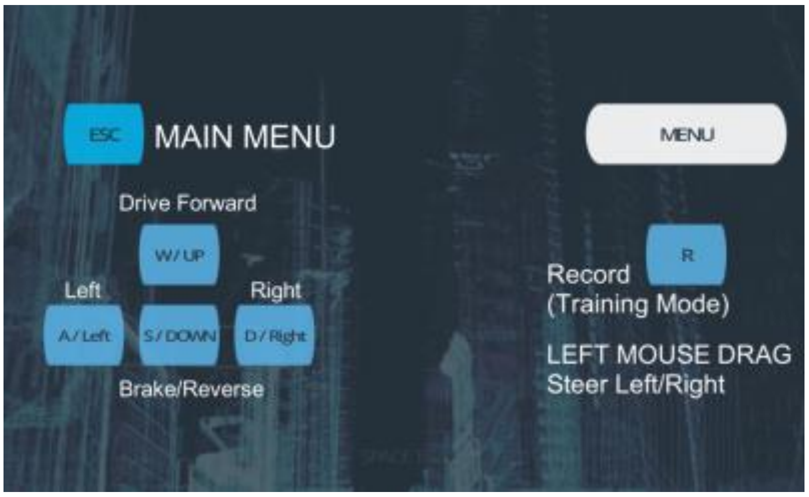


Figure 5.3. Controls Configuration

As seen in the main menu (Figure 5.2.), there are 2 different maps in the simulation. The first map has a simpler and less complex structure and the second map has more corners and a more difficult to drive map structure.

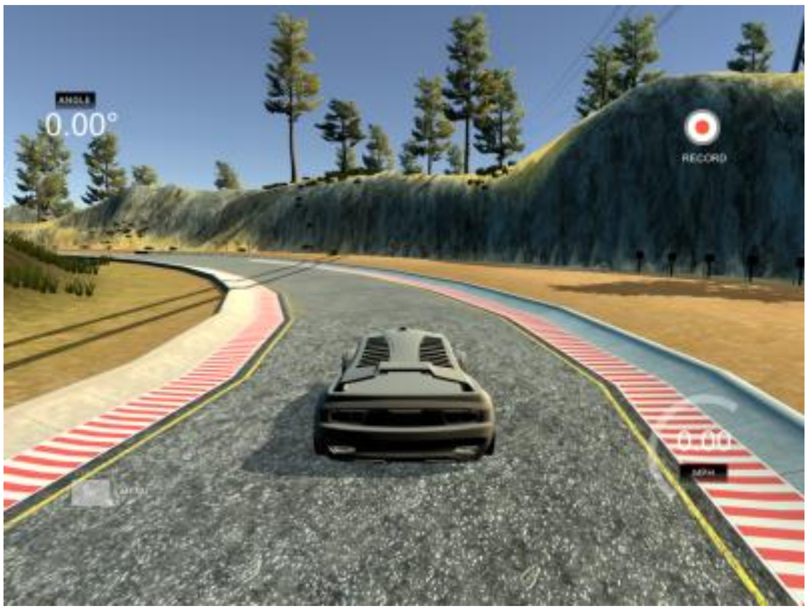


Figure 5.4. Track-1

Map 2 is a more difficult map to drive, with steep slopes, sharp curves, areas of shadow and different light brightnesses.

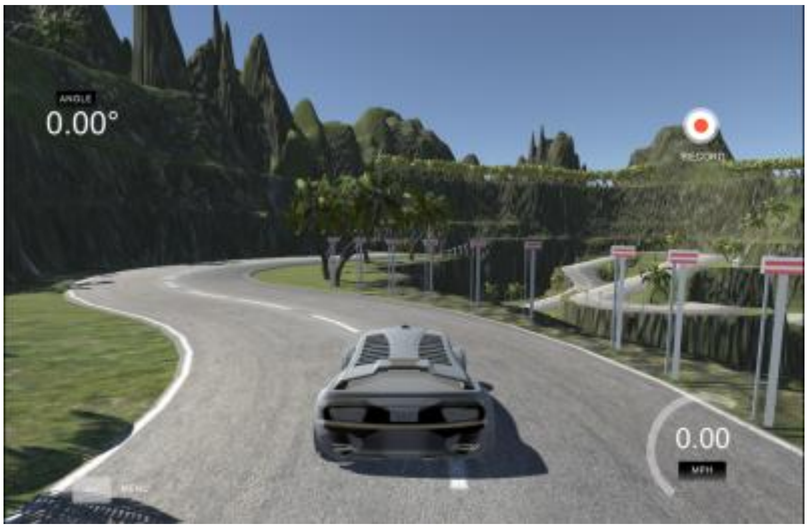


Figure 5.5. Track-2

There are 2 different modes to use the car, the first is the training mode and the second is the autonomous mode. (Figure 5.2.)

In training mode, it is necessary to record while driving the car to prepare dataset. For this, the red icon in the upper right allows us to record. (Figure 5.5.)

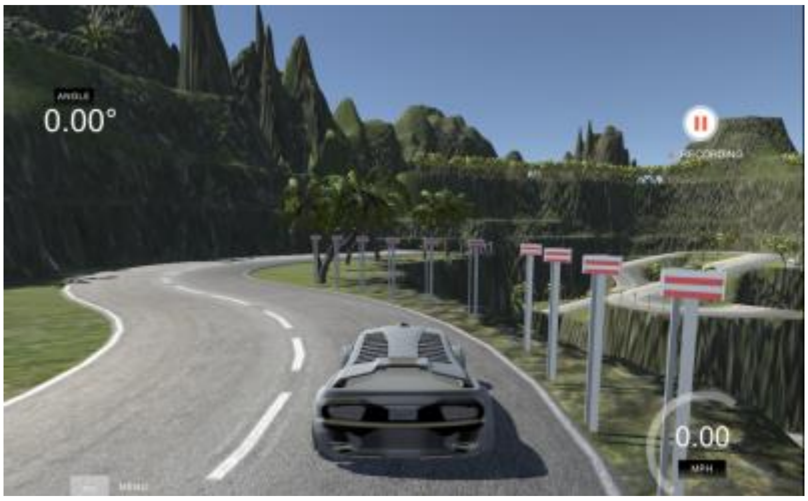


Figure 5.6. Recording in training mode

In autonomous mode, it is a simulator mode in which the car drives itself without any human intervention after training the model.

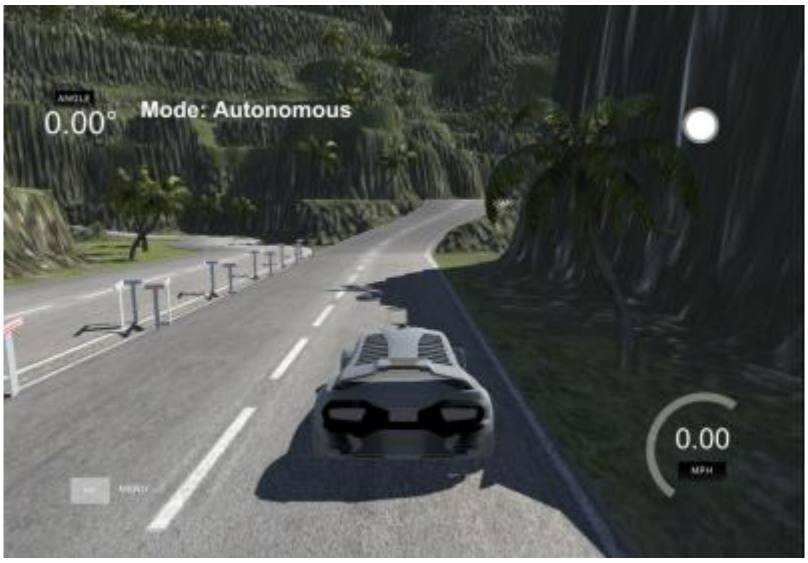


Figure 5.7. Autonomous mode

The simulation's recording feature while driving makes it very easy to create a dataset to be used in the problem.

Some features of the simulation:

- The simulator can take continuous screenshots of driving features from 3 different cameras (center, left , right).

- The stream of images is captured and the captured images can be saved in any location on the disk. Captured images are tagged from which camera they were captured.

- The simulator creates the driving_log.csv file with the image dataset. This file contains data that corresponds to the image, including Steering angle, throttle, brakes and speed of the car.

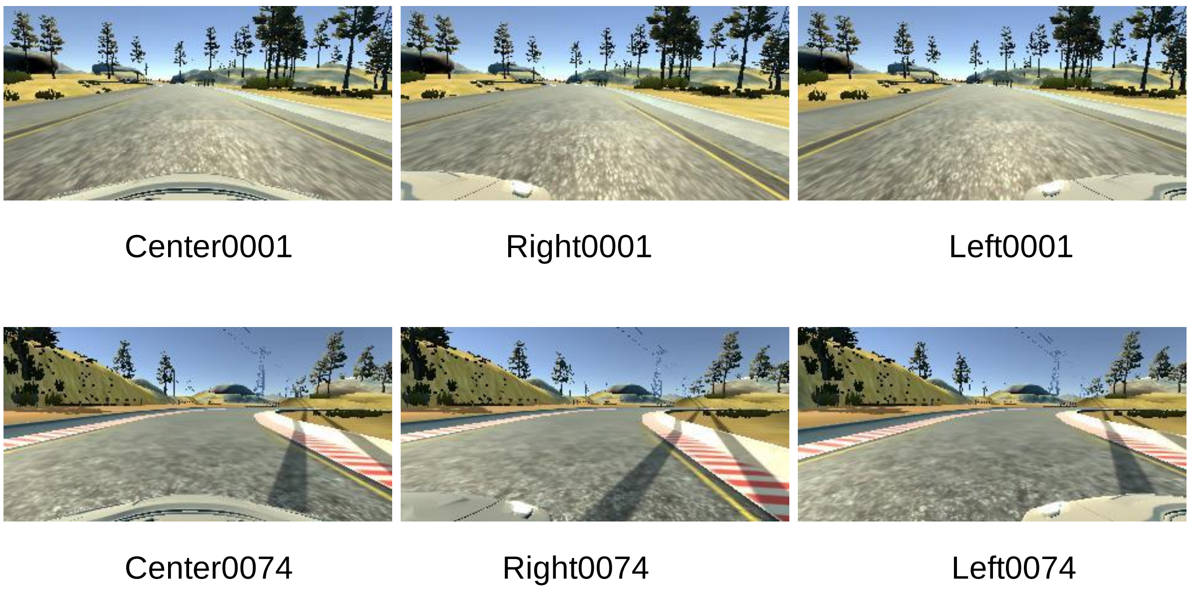
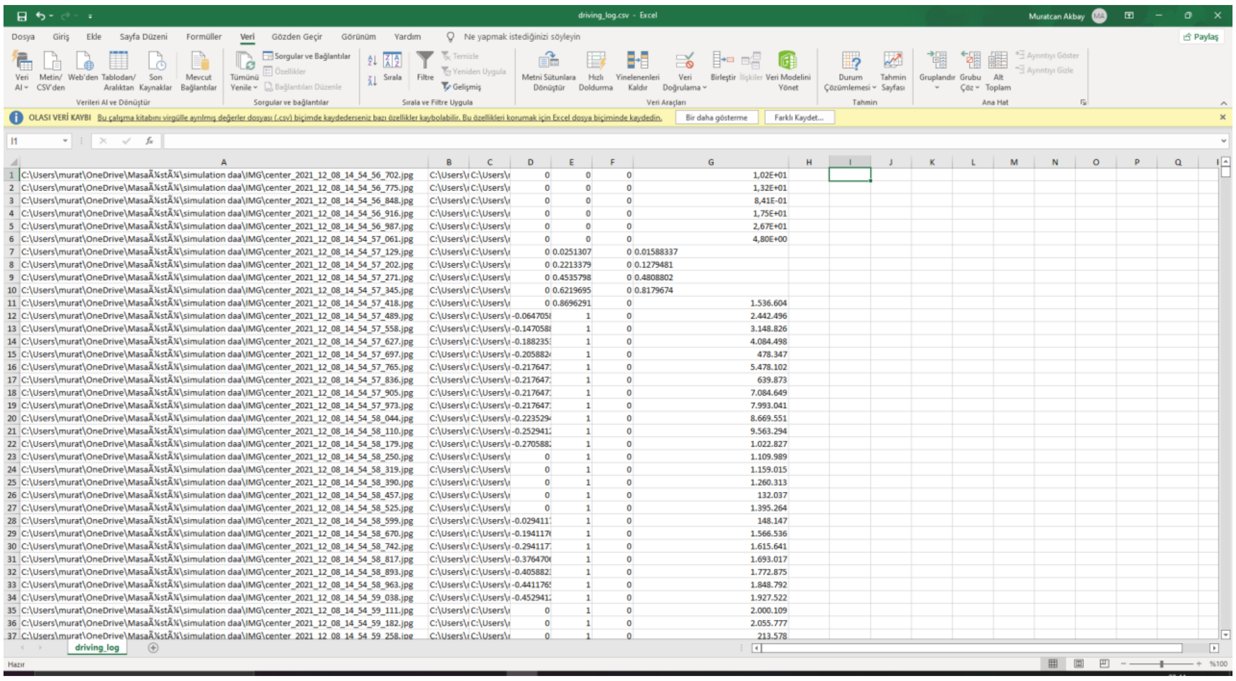


Figure 5.8. Dataset Sample

Some images in the dataset are shown in Figure 5.8. .



	A	B	C	D	E	F	G
1	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_56_702.jpg	C:\Users\	C:\Users\	0	0	0	1,02E+01
2	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_56_775.jpg	C:\Users\	C:\Users\	0	0	0	1,32E+01
3	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_56_848.jpg	C:\Users\	C:\Users\	0	0	0	8,41E-01
4	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_56_916.jpg	C:\Users\	C:\Users\	0	0	0	1,75E+01
5	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_56_987.jpg	C:\Users\	C:\Users\	0	0	0	2,67E+01
6	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_57_061.jpg	C:\Users\	C:\Users\	0	0	0	4,80E+00
7	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_57_129.jpg	C:\Users\	C:\Users\	0	0.0251307	0	0.01588337
8	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_57_202.jpg	C:\Users\	C:\Users\	0	0.2213379	0	0.1279481
9	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_57_271.jpg	C:\Users\	C:\Users\	0	0.4535798	0	0.4808802
10	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_57_345.jpg	C:\Users\	C:\Users\	0	0.6219695	0	0.8179674
11	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_57_418.jpg	C:\Users\	C:\Users\	0	0.8696291	0	1.536.604
12	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_57_489.jpg	C:\Users\	C:\Users\	-0.0647058	1	0	2.442.496
13	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_57_558.jpg	C:\Users\	C:\Users\	-0.1470588	1	0	3.148.826
14	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_57_627.jpg	C:\Users\	C:\Users\	-0.1882353	1	0	4.084.498
15	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_57_697.jpg	C:\Users\	C:\Users\	-0.2058824	1	0	478.347
16	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_57_765.jpg	C:\Users\	C:\Users\	-0.2176471	1	0	5.478.102
17	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_57_836.jpg	C:\Users\	C:\Users\	-0.2176471	1	0	639.873
18	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_57_905.jpg	C:\Users\	C:\Users\	-0.2176471	1	0	7.084.649
19	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_57_973.jpg	C:\Users\	C:\Users\	-0.2176471	1	0	7.993.041
20	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_58_044.jpg	C:\Users\	C:\Users\	-0.2235294	1	0	8.669.551
21	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_58_110.jpg	C:\Users\	C:\Users\	-0.2529412	1	0	9.563.294
22	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_58_179.jpg	C:\Users\	C:\Users\	-0.2705882	1	0	1.022.827
23	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_58_250.jpg	C:\Users\	C:\Users\	0	1	0	1.109.989
24	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_58_319.jpg	C:\Users\	C:\Users\	0	1	0	1.159.015
25	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_58_390.jpg	C:\Users\	C:\Users\	0	1	0	1.260.313
26	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_58_457.jpg	C:\Users\	C:\Users\	0	1	0	132.037
27	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_58_525.jpg	C:\Users\	C:\Users\	0	1	0	1.395.264
28	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_58_599.jpg	C:\Users\	C:\Users\	-0.0294117	1	0	148.147
29	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_58_670.jpg	C:\Users\	C:\Users\	-0.1941176	1	0	1.566.536
30	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_58_742.jpg	C:\Users\	C:\Users\	-0.2941177	1	0	1.615.641
31	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_58_817.jpg	C:\Users\	C:\Users\	-0.3764706	1	0	1.693.017
32	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_58_893.jpg	C:\Users\	C:\Users\	-0.4058823	1	0	1.772.875
33	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_58_963.jpg	C:\Users\	C:\Users\	-0.4411765	1	0	1.848.792
34	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_59_038.jpg	C:\Users\	C:\Users\	-0.4529412	1	0	1.927.522
35	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_59_111.jpg	C:\Users\	C:\Users\	0	1	0	2.000.109
36	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_59_182.jpg	C:\Users\	C:\Users\	0	1	0	2.055.777
37	C:\Users\murat\OneDrive\MasaÃ¼stÃ¼\simulation daa\IMG\center_2021_12_08_14_54_59_258.jpg	C:\Users\	C:\Users\	0	1	0	213.578

Figure 5.9. driving_log.csv

Column 1: Shows the path of the image "center" in the dataset.

Column 2: Shows the path of the image "right" in the dataset.

Column 3: Shows the path of the image "left" in the dataset.

Column 4: Shows the steering angle. If the steering angle value is 0, it indicates that it is straight. Negative values of the steering angle indicate left turns. Positive values of the steering angle indicate right turns.

Column 5: Shows the acceleration value.

Column 6: Shows the deceleration value.

Column 7: Shows the instantaneous speed value.

4. ARCHITECTURE

First Model (NVIDIA)

Nvidia has released a model for self-driving cars [6]. This model architecture is built and trained based on data from real cars.

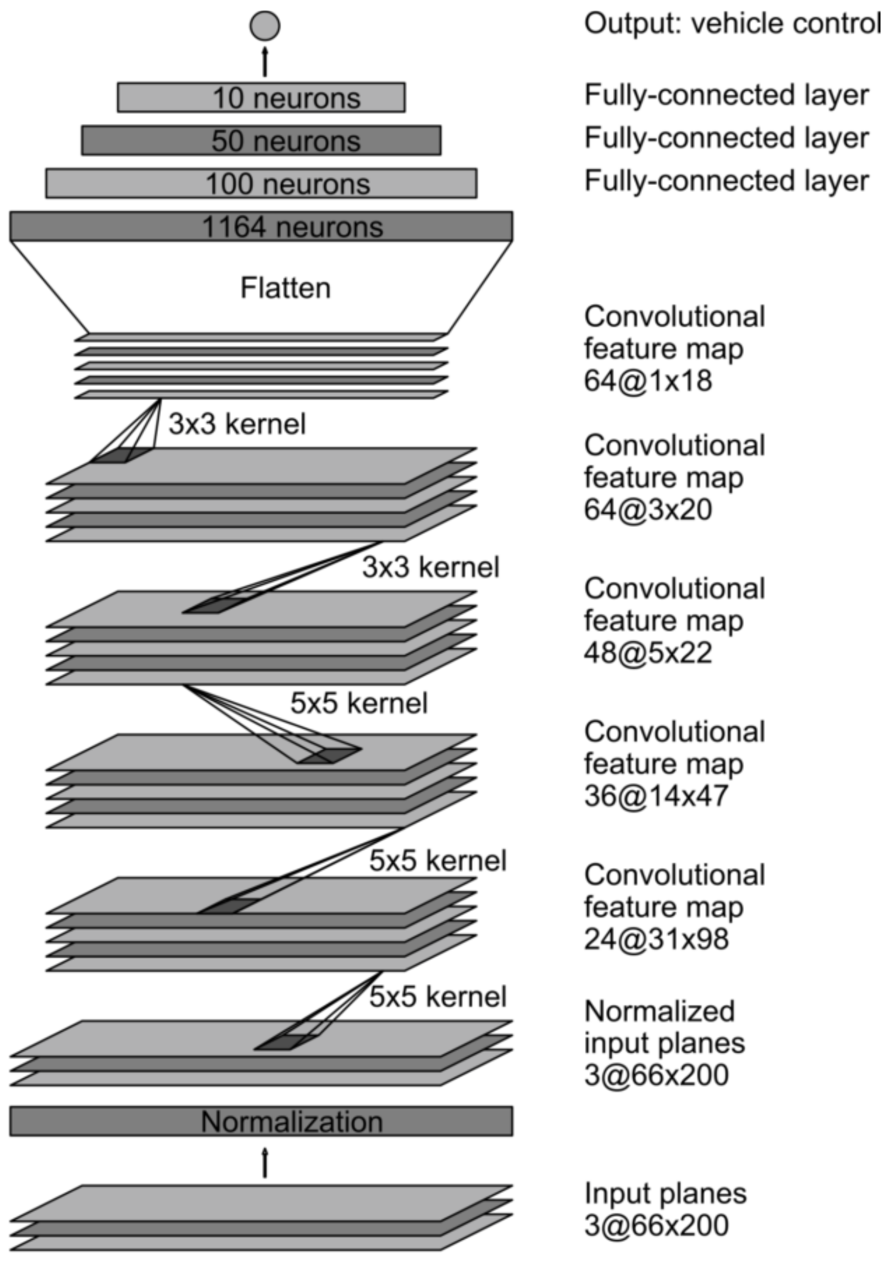
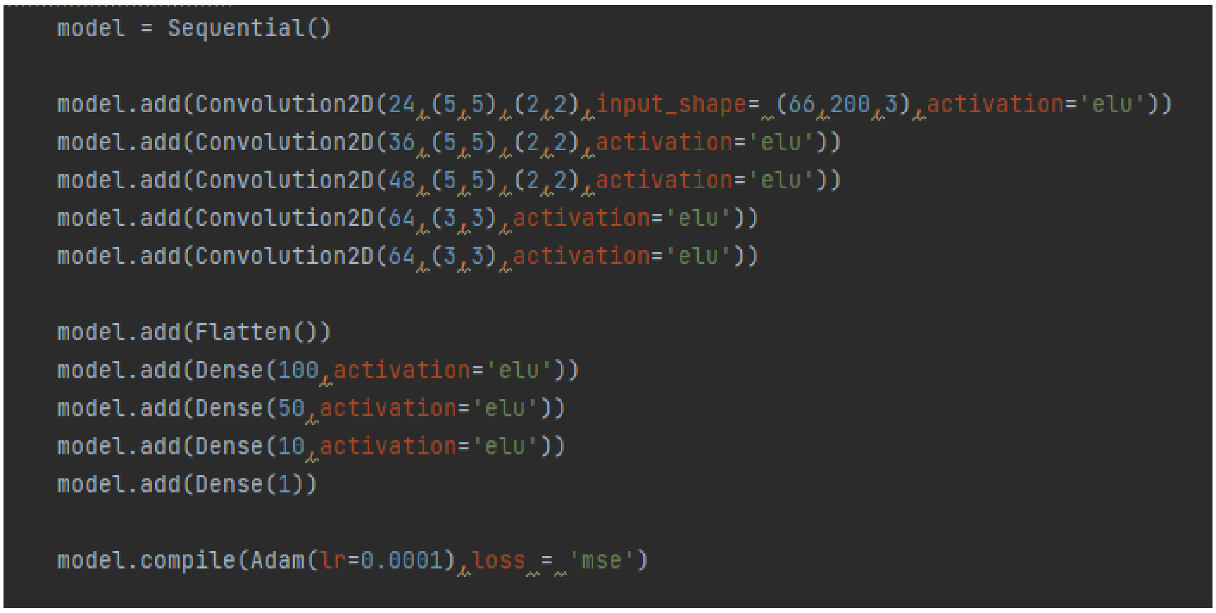
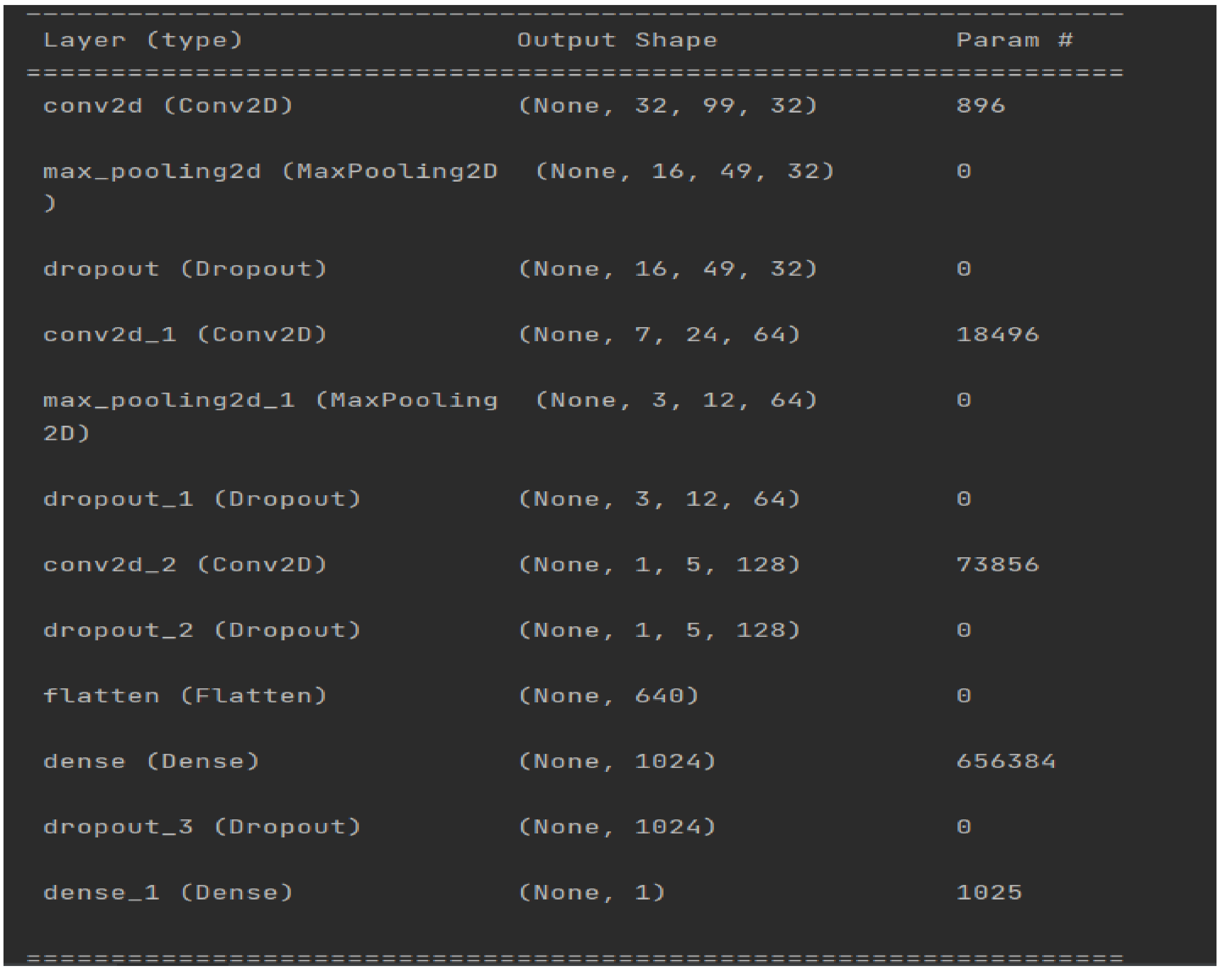


Figure 6.1. CNN architecture. The network has about 27 million connections and 250 thousand parameters.

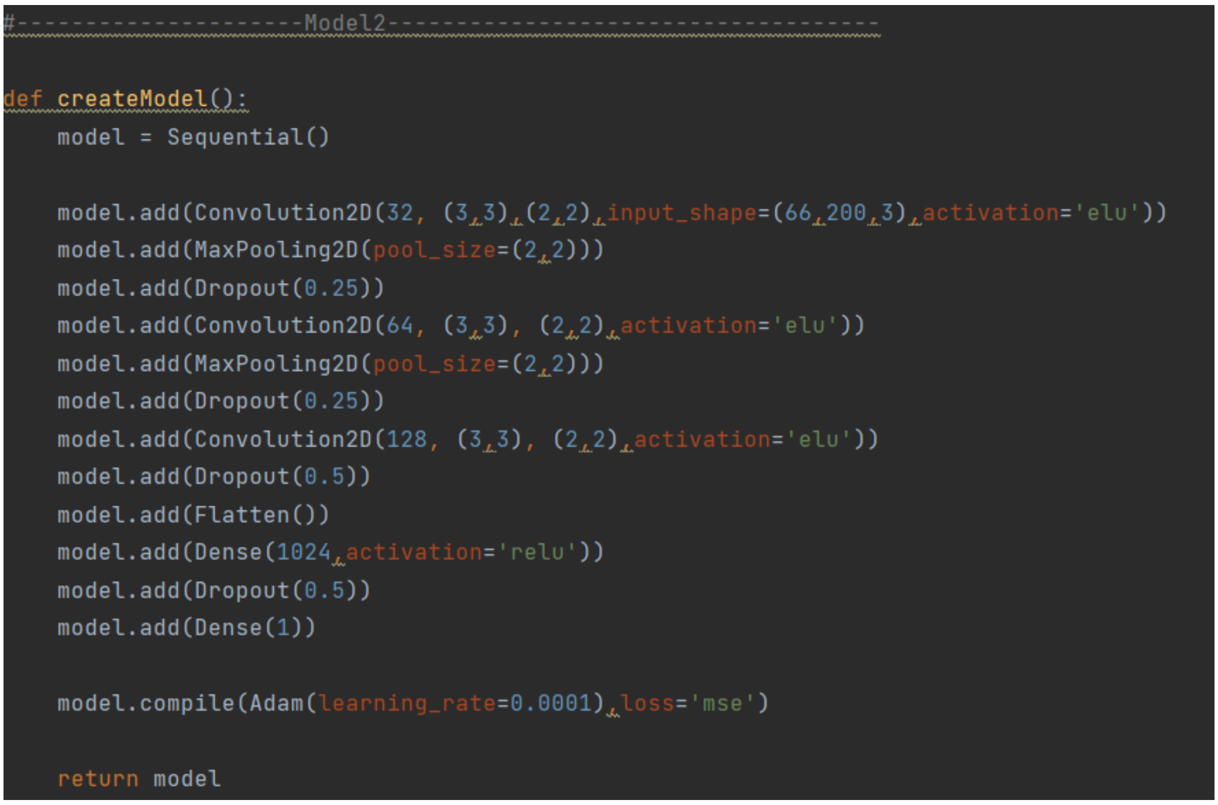


```
model = Sequential()

model.add(Convolution2D(24,(5,5),(2,2),input_shape= (66,200,3),activation='elu'))
model.add(Convolution2D(36,(5,5),(2,2),activation='elu'))
model.add(Convolution2D(48,(5,5),(2,2),activation='elu'))
model.add(Convolution2D(64,(3,3),activation='elu'))
model.add(Convolution2D(64,(3,3),activation='elu'))

model.add(Flatten())
model.add(Dense(100,activation='elu'))
model.add(Dense(50,activation='elu'))
model.add(Dense(10,activation='elu'))
model.add(Dense(1))

model.compile(Adam(lr=0.0001),loss = 'mse')
```

Second Model (AlexNet)

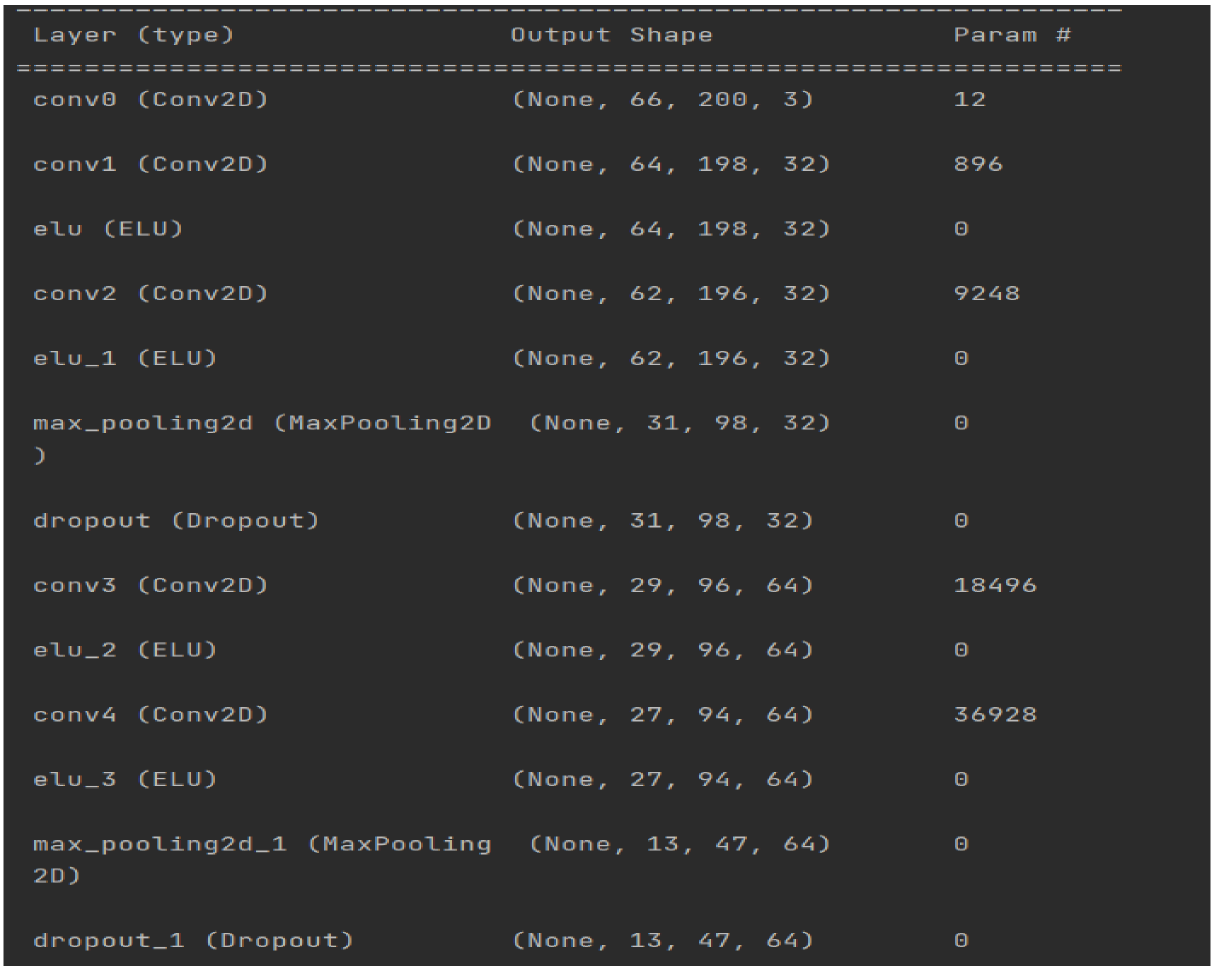


```
 _________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 99, 32)        896

 max_pooling2d (MaxPooling2D  (None, 16, 49, 32)       0
 )

 dropout (Dropout)           (None, 16, 49, 32)        0

 conv2d_1 (Conv2D)           (None, 7, 24, 64)         18496

 max_pooling2d_1 (MaxPooling  (None, 3, 12, 64)        0
 2D)

 dropout_1 (Dropout)         (None, 3, 12, 64)         0

 conv2d_2 (Conv2D)           (None, 1, 5, 128)         73856

 dropout_2 (Dropout)         (None, 1, 5, 128)         0

 flatten (Flatten)           (None, 640)               0

 dense (Dense)               (None, 1024)              656384

 dropout_3 (Dropout)         (None, 1024)              0

 dense_1 (Dense)             (None, 1)                 1025

=================================================================
```

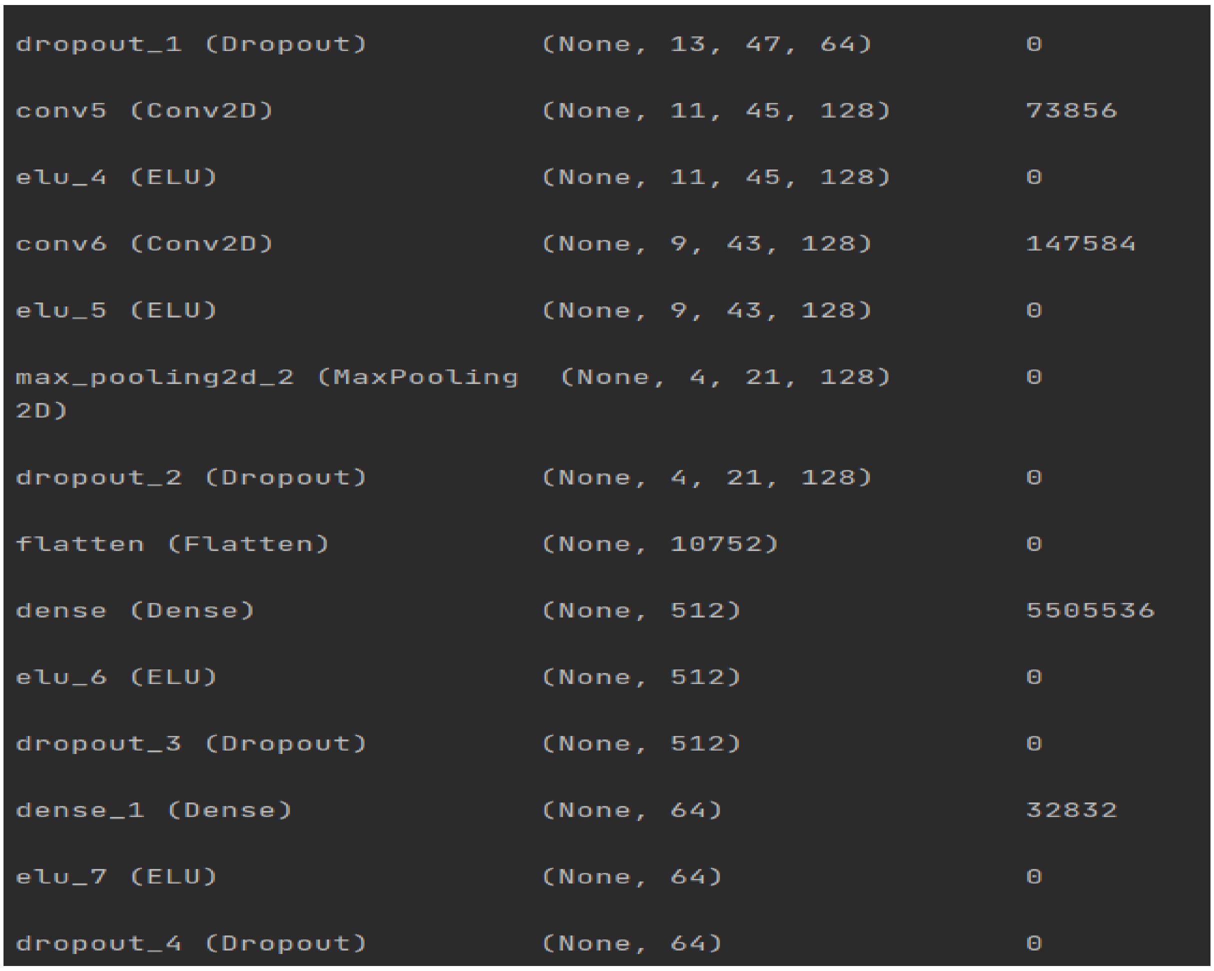


```
#---------------------Model2-----------------------------------

def createModel():
    model = Sequential()

    model.add(Convolution2D(32, (3,3),(2,2),input_shape=(66,200,3),activation='elu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.25))
    model.add(Convolution2D(64, (3,3), (2,2),activation='elu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.25))
    model.add(Convolution2D(128, (3,3), (2,2),activation='elu'))
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(1024,activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(1))

    model.compile(Adam(learning_rate=0.0001),loss='mse')

    return model
```

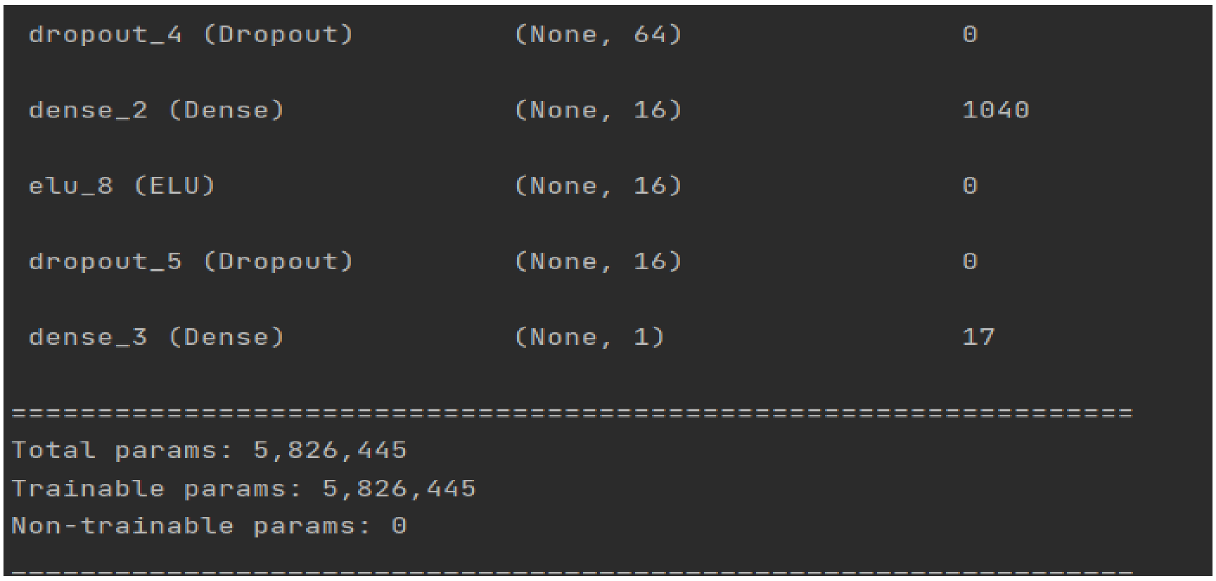
Third Model (VGG16)



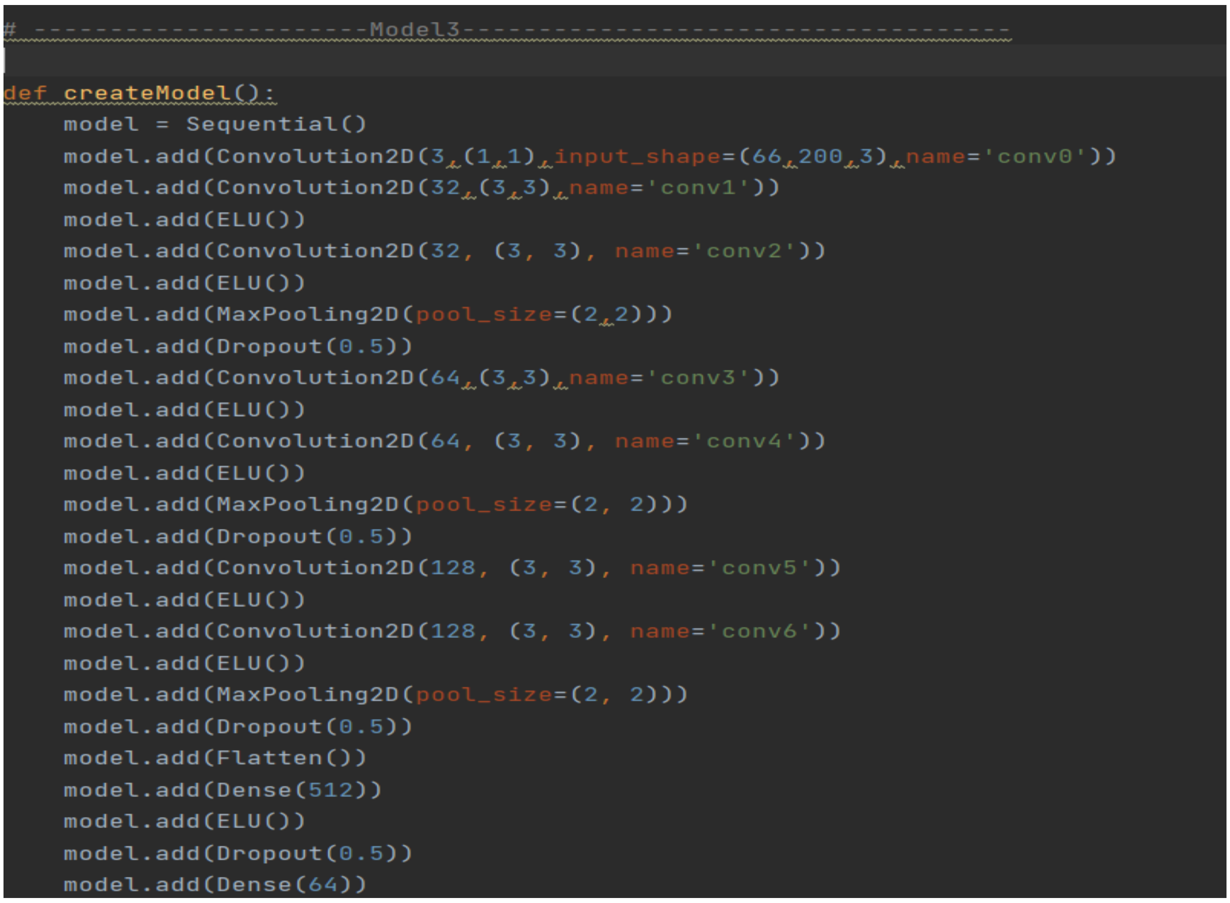
```
 ____________________________________________________________________
  Layer (type)                Output Shape              Param #
 ====================================================================
  conv0 (Conv2D)              (None, 66, 200, 3)        12

  conv1 (Conv2D)              (None, 64, 198, 32)       896

  elu (ELU)                   (None, 64, 198, 32)       0

  conv2 (Conv2D)              (None, 62, 196, 32)       9248

  elu_1 (ELU)                 (None, 62, 196, 32)       0

  max_pooling2d (MaxPooling2D  (None, 31, 98, 32)       0
  )

  dropout (Dropout)           (None, 31, 98, 32)        0

  conv3 (Conv2D)              (None, 29, 96, 64)        18496

  elu_2 (ELU)                 (None, 29, 96, 64)        0

  conv4 (Conv2D)              (None, 27, 94, 64)        36928

  elu_3 (ELU)                 (None, 27, 94, 64)        0

  max_pooling2d_1 (MaxPooling  (None, 13, 47, 64)       0
  2D)

  dropout_1 (Dropout)         (None, 13, 47, 64)        0
```



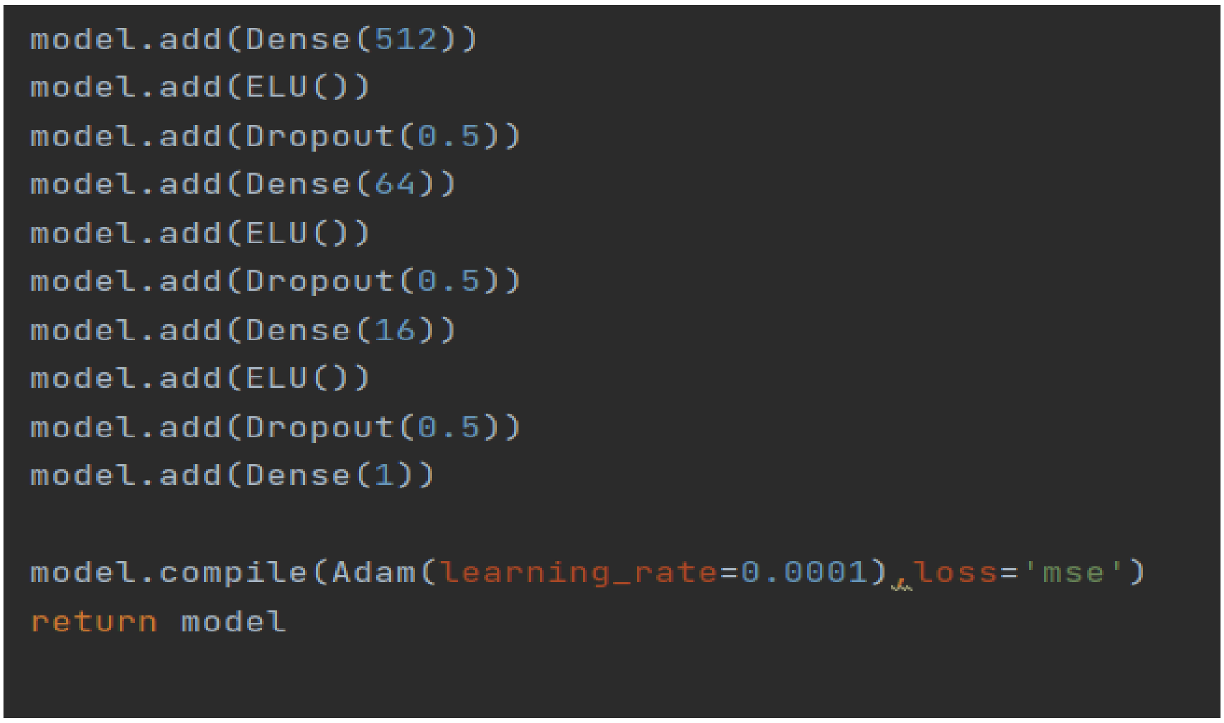
```
dropout_1 (Dropout)           (None, 13, 47, 64)        0

conv5 (Conv2D)                (None, 11, 45, 128)       73856

elu_4 (ELU)                   (None, 11, 45, 128)       0

conv6 (Conv2D)                (None, 9, 43, 128)        147584

elu_5 (ELU)                   (None, 9, 43, 128)        0

max_pooling2d_2 (MaxPooling   (None, 4, 21, 128)        0
2D)

dropout_2 (Dropout)           (None, 4, 21, 128)        0

flatten (Flatten)             (None, 10752)             0

dense (Dense)                 (None, 512)               5505536

elu_6 (ELU)                   (None, 512)               0

dropout_3 (Dropout)           (None, 512)               0

dense_1 (Dense)               (None, 64)                32832

elu_7 (ELU)                   (None, 64)                0

dropout_4 (Dropout)           (None, 64)                0
```



```
 dropout_4 (Dropout)          (None, 64)                0

 dense_2 (Dense)              (None, 16)                1040

 elu_8 (ELU)                  (None, 16)                0

 dropout_5 (Dropout)          (None, 16)                0

 dense_3 (Dense)              (None, 1)                 17

=================================================================
Total params: 5,826,445
Trainable params: 5,826,445
Non-trainable params: 0
_________________________________________________________________
```



```
# ----------------------Model3-----------------------------------

def createModel():
    model = Sequential()
    model.add(Convolution2D(3,(1,1),input_shape=(66,200,3),name='conv0'))
    model.add(Convolution2D(32,(3,3),name='conv1'))
    model.add(ELU())
    model.add(Convolution2D(32, (3, 3), name='conv2'))
    model.add(ELU())
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.5))
    model.add(Convolution2D(64,(3,3),name='conv3'))
    model.add(ELU())
    model.add(Convolution2D(64, (3, 3), name='conv4'))
    model.add(ELU())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.5))
    model.add(Convolution2D(128, (3, 3), name='conv5'))
    model.add(ELU())
    model.add(Convolution2D(128, (3, 3), name='conv6'))
    model.add(ELU())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(512))
    model.add(ELU())
    model.add(Dropout(0.5))
    model.add(Dense(64))
```



```
model.add(Dense(512))
model.add(ELU())
model.add(Dropout(0.5))
model.add(Dense(64))
model.add(ELU())
model.add(Dropout(0.5))
model.add(Dense(16))
model.add(ELU())
model.add(Dropout(0.5))
model.add(Dense(1))

model.compile(Adam(learning_rate=0.0001),loss='mse')
return model
```

Convolutional Layer: Convolutional (convolutional layer) is the first layer that handles the image in CNN algorithms. As it is known, images are actually matrices consisting of pixels with certain values in them. In the convolution layer, a filter smaller than the original image size hovers over the image and tries to capture certain features from these images.

Pooling (Downsampling) Layer: Reduces the number of weights. In this way, both the required processing power is reduced and the unnecessary features that are caught are ignored and more important features are focused on.

Flattening Layer: It is used to flatten data in matrix form.

Fully-Connected Layer: At the nodes in the layers, the features are kept and the learning process is entered by changing the weight and bias.

Dropout Layer : The Dropout layer **randomly sets input units to 0 with a frequency of rate at each step during training time**, which helps prevent overfitting.

Dense Layer: Dense Layer is **simple layer of neurons in which each neuron receives input from all the neurons of previous layer**, thus called as dense.

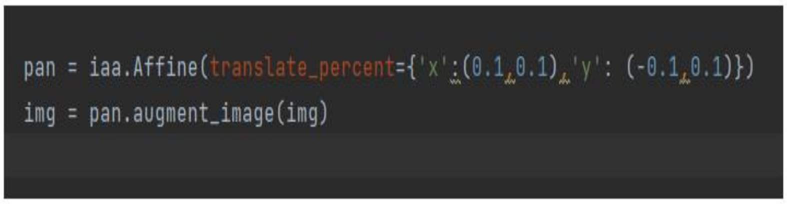
5. AUGMENTATION AND IMAGE PRE-PROCESSING

The car has been trained only with Track_1 data and is expected to drive successfully on a different map which is Track_2, using this data. However, Track_2 contains different conditions than Track_1 and these conditions are not included in the data we collect.

This process is valid for autonomous cars used in real life. Cars are trained using certain data, but this data is not collected by training in all road conditions in the world. This process requires a lot of data, which can be time-consuming and storage-intensive. To solve this problem we use image processing and augmentation techniques which will be discussed in this chapter.

Shift

The images in the dataset are shifted by a small amount. This contributes to the data diversity by creating the new images which are located differently then the original images. The code snippet and example of pan is given below.



```
pan = iaa.Affine(translate_percent={'x':(0.1,0.1),'y': (-0.1,0.1)})
img = pan.augment_image(img)
```

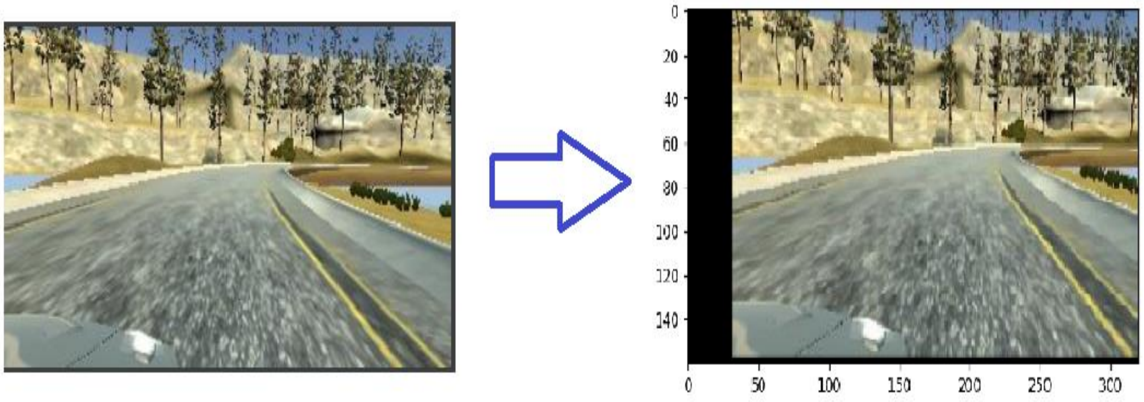
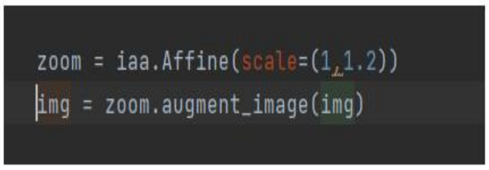


Figure 7.1. Shift image

Zoom

The images in the dataset are zoomed in by a small amount. This contributes to the data diversity by creating the new images which are different then the original ones. The snippet of code and transformation of an image after zooming is given below.



```
zoom = iaa.Affine(scale=(1,1.2))
img = zoom.augment_image(img)
```

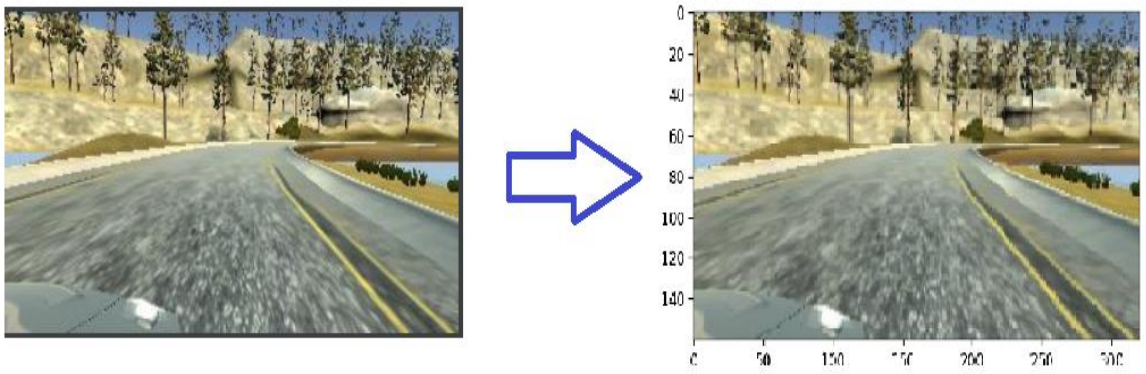
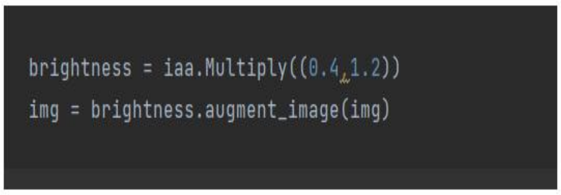


Figure 7.2. Zoom image

Brightness

The images in the dataset are increased brightness by a small amount. This contributes to the data diversity by creating the new images which has different weather conditions like cloudy, sunny and lowlight conditions. The code snippet and increase of brightness example is given below.



```
brightness = iaa.Multiply((0.4,1.2))
img = brightness.augment_image(img)
```

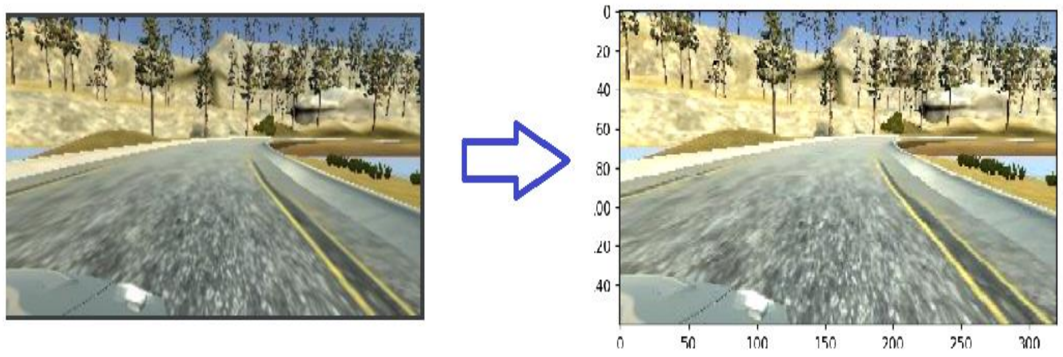
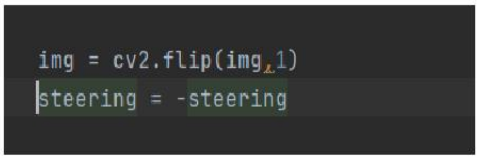


Figure 7.3 Random Brightness

Flip

The images in the dataset flipped horizontally. This contributes to the data diversity by creating the new images which has similar kinds of turns on opposite sides too. The snippet of code and transformation of an image after flipping it is given below.



```
img = cv2.flip(img,1)
steering = -steering
```

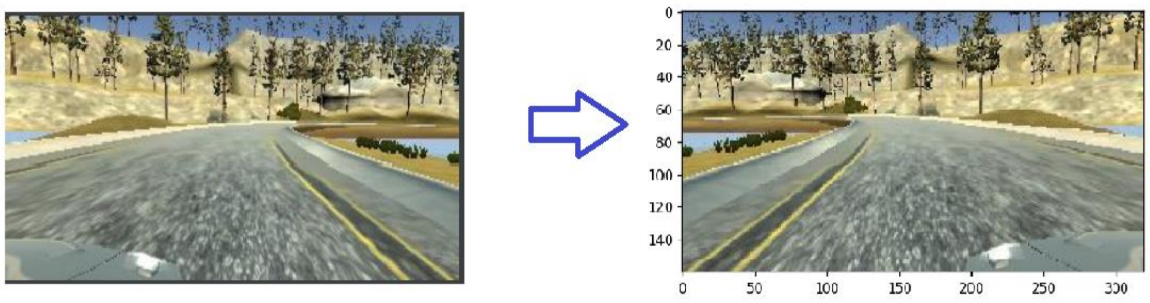
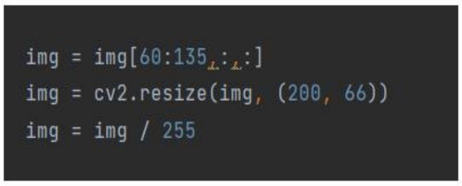


Figure 7.4. Flip image

Crop

The images in the dataset have relevant features in the lower part where the road is visible. The external environment above a certain image portion will never be used to determine the output and thus can be cropped. The snippet code and example of cropping is given below.



```
img = img[60:135,:,:]
img = cv2.resize(img, (200, 66))
img = img / 255
```

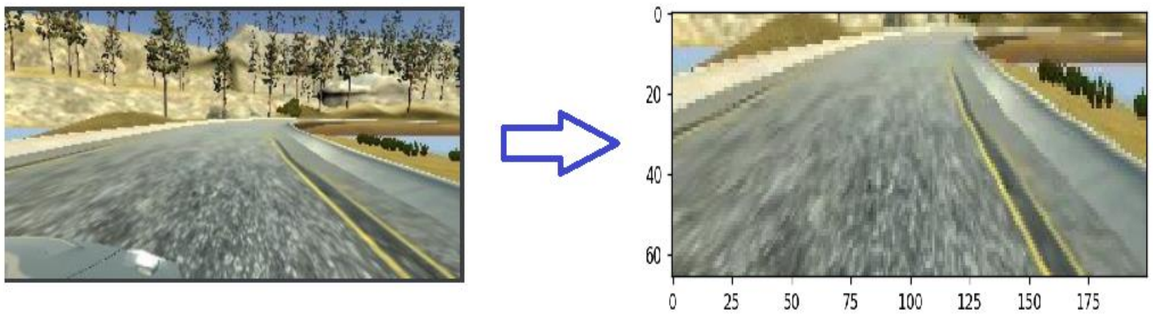
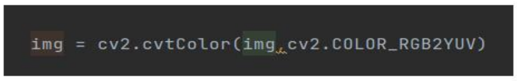


Figure 7.5 Crop image

Color Change

To better define where the lane lines of the road and the path is ,changing colors (RGB to YUV) helps us. After the changing colors the new image and snippet code is given below.



```
img = cv2.cvtColor(img,cv2.COLOR_RGB2YUV)
```

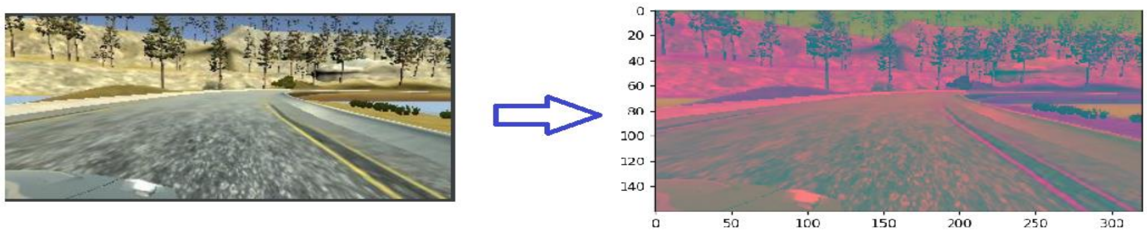
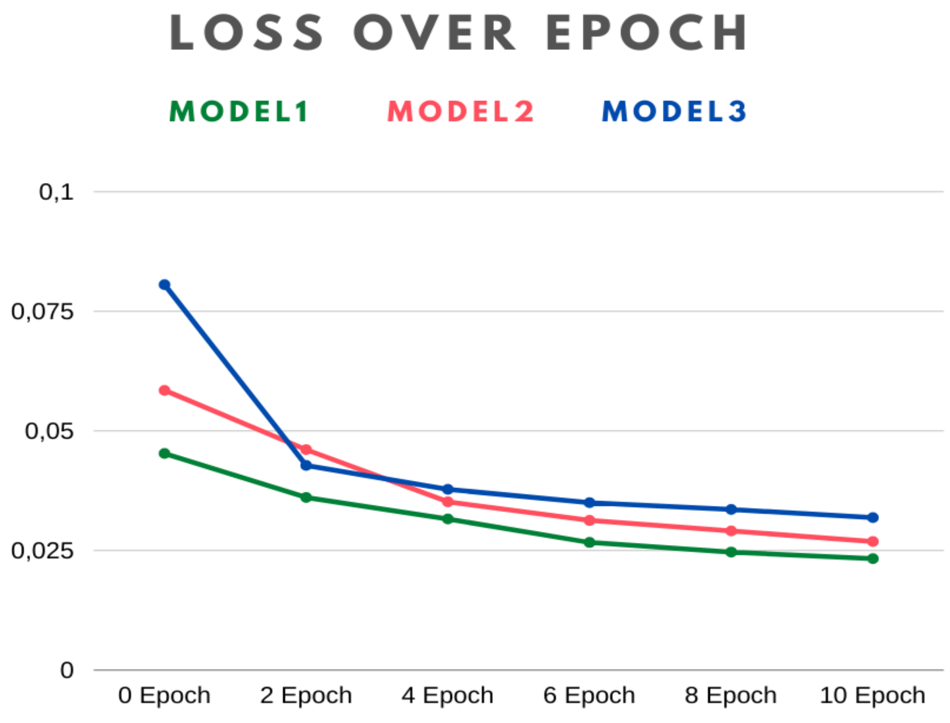


Figure 7.6 Color change

6.RESULT

Our aim in this project is to examine the behavioral movements on the tracks where these cars have never met and trained, other than autonomous car driving, and find solutions accordingly.

One of these solutions is to differentiate and multiply the dataset by using augmentation techniques with limited dataset and to diversify the dataset so that the model can give better results in other tracks.



	True Case	False Case
Model1	65	35
Model2	35	65

	True Case	False Case
Model2	13	87
Model3	87	13

	True Case	False Case
Model1	33	67
Model3	67	33

To determine which model is more successful in total by assigning '1' for the model with a closer value to the performance of different artificial intelligence models compared to the distance of the car to the expected steering angle values for a safe drive, and '0' for the model with a farther value.

Track 1 Performance :

In track-1, the trained models complete the track successfully. The applied augmentation techniques gave successful results. If the dataset is increased and diversified in order to increase the performance of the car on track-1, the performance of the model on the track will improve.

Track 2 Performance:

In a Track 2 is a track that models have not trained and encountered before. The 2nd track has more difficult conditions than the 1st track. There are more canopy areas, slopes, ramps and bends. All 3 trained models performed different performances.

As a result of Mcnemar tests, the comparison of the models and the probability of successful results were examined.

According to Mcnemar tests, model 3 gives the best results and model 2 has the worst performance.

To increase the performance of these models, it is necessary to increase the dataset, use better augmentation techniques and develop model architectures.

7.CONCLUSION

Our project is a concept that needs to be addressed, taking into account our national and social planning and evaluation of autonomous vehicle simulation. By using a simulator with tracks with more professional conditions (different vehicles, traffic signs, etc.) and with more up-to-date and structural artificial intelligence models and longer training processes, the closest possible simulation can be achieved in education.

Many new developments and experiments are taking place in this field, and this project is important in terms of benefiting the experiments. New maps, objects, other transportation vehicles, traffic signs and unexpected conditions can be integrated into the simulation in order to provide more benefits to these experiments and to provide many advantages for the purpose of the project by providing the closest simulation environment to reality.

However, in order to achieve success in an advanced simulation environment, much more complex artificial intelligence models and very long training periods may be required. Our project can be considered as a step in this respect.

REFERENCES

[1] Keshav Bimbraw Mechanical Engineering Department, Thapar University, P.O. Box 32, Patiala, Punjab

[2] https://www.investopedia.com/articles/investing/052014/how-googles-selfdriving-carwill-change-everything.asp

[3] California 2016. Autonomous Vehicle Disengagement Reports. https://www.dmv.ca.gov/portal/dmv/detail/vr/autonomous/disengagement_
report_2016. (2016).

[4] Oliver Cameron, “Challenge #2: Using Deep Learning to Predict Steering Angles”, Published on 11 Aug 2016, https://medium.com/udacity/challenge-2-using-deeplearning-to-predict-steering-angles-f42004a36ff3,

[5] Deep Learning Nedir? - Radu Raicea

https://nyilmazsimsek.medium.com/derin-%C3%B6%C4%9Frenme-deep-learning-nedir-ve-nas%C4%B1l-%C3%A7al%C4%B1%C5%9F%C4%B1r-2d7f5850782

[6] Mariusz Bojarski, “End-to-End Deep Learning for Self-Driving Cars”, Published on 17 Aug, 2016, https://devblogs.nvidia.com/deep-learning-self-driving-cars

[7] Dmytro Nasyrov “Behavioral Cloning. NVidia Neural Network in Action.” https://towardsdatascience.com/behavioral-cloning-project-3- 6b7163d2e8f9

[8] Ivan Kazakov “Vehicle Detection and Tracking”, Published on 14 May 2017, https://towardsdatascience.com/vehicle-detection-and-tracking-44b851d70508