# **P3 Behavioural Cloning**

The main objective of this project is to develop a model that makes the car drive in a test circuit by itself. This is achieved in this project by using a method called Behavioural Cloning. We will see about it in detail in this README file.

## Files in this Repo

* model.py – consists of the code for importing the frames, pre-processing the frames, training the mode and saving the model
* model.ib – contains the training model weights which can be used later in the simulation while driving
* drive.py – script for driving the car in the simulator
* TrainingData/ - folder which contains the extracted frames while driving. These frames are used for training the model. Each frame is of size 320 x 160 pixel
* model.ipynb – Jupyter notebook containing the same code as model.py

## Properties of the Training Set

* The training images used for training the model can be found in the folder /TrainingData/IMG/ . Here there are three types of images. They are as follows,
  1. The images starting with the name “centre” are taken directly in front of the car.
  2. The images starting with the name “left” are taken slightly facing the left side
  3. The images starting with the name “right” are taken slightly facing the right side
* All the images are of type JPG
* The images are of size 320 x 120 pixels
* The color pattern oft he images is RGB
* The labels for the images can be found in the file TrainingData/driving\_log.csv
* The labels are steering angle, throttle, speed, brake, etc.

## Sample Frame



Here is this image,

1. Camera location is Left

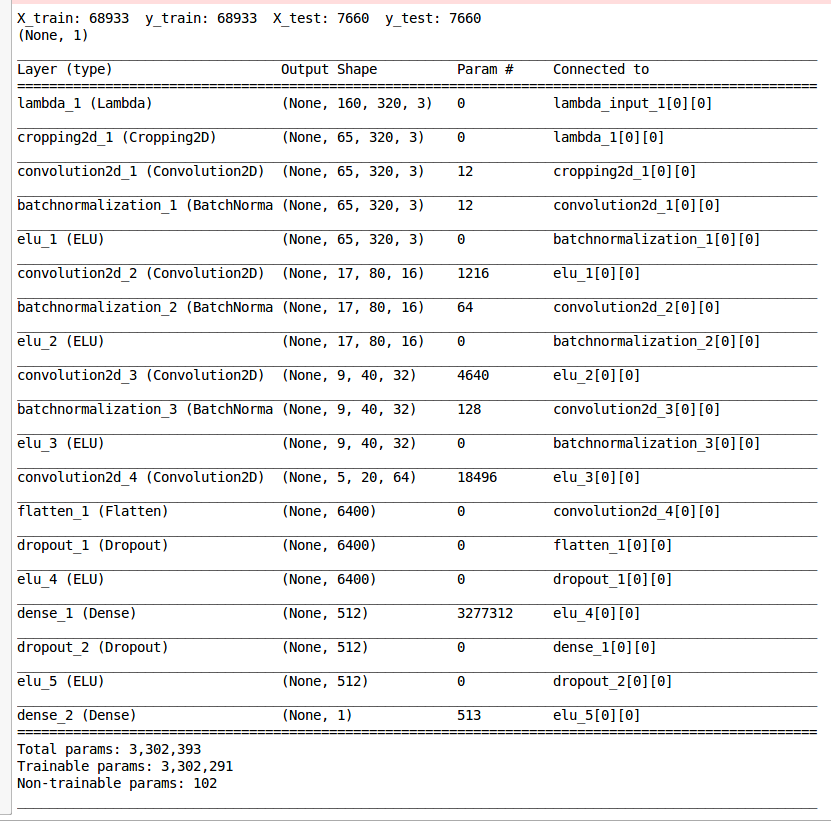
2. Steering angle is -0.1416309

3. Throttle is 1

4. Brake is 0.

## Model Architecture

My model architecture is as follows.



Here the first two (Lambda\_1 and cropping2d\_1) are the preprocessing steps.

The first step is normnalization of the values using the formula x/255 – 0.5 and in the second step the images are cropped so that only the road portion is in the image and the unwanted scenaries and the bonnet of the car are removed.

The Arcitecture consists of 4 convolution layers. Dropout is used for generalization of the model and to avoid overfitting.

## Final Model and its Training Data

I collected almost 75000 frames. I divided them into training and validation sets. I also used a shuffle to make sure they are fed in a different order each time for training. The training frames are almore 69000 and the validation framesa are 7600. After training the model, the model was able to drive the car around the circuit without going outside the test track. I did 10 epochs with batch size 240.