Project- Behavioral Cloning

1. Major Steps:
   - ✓ Data collection by manually driving the vehicle in simulator.
     - Driving in the direction to avoid one sided bias.
     - Speed is not the consideration.
     - Also, the model needs to be trained for adverse situation, so its ok to challenge it for extreme situation.
     - All these data is stored in image format, also the driving behavior is logged in an excel file with camera and steering, drive and pitch data.
   - ✓ Build, a convolution neural network in Keras that predicts steering angles from images.
   - ✓ Train and validate the model with a training and validation set.
   - ✓ Testing the model in autonomous mode in simulator
     - Vehicle should successfully be able to drive around the track one time without leaving the road.
     - This data will be recorded in terms of images with time stamps.
   - ✓ Generating a video file with the time stamped image data (from previous step).

2. Submission file includes (majorly):
   - ✓ drive.py
   - ✓ model.py
   - ✓ model3.h5
   - ✓ Data: Training data 'Data', which consist 'IMG' and 'driving_log.csv' files.

     Udacity encourages students to collect their own data. They also emphasis on recovery data, as that will make the model to tackle in adverse scenarios. Therefore the data was collected from the starting with special emphasis on steering the car back towards the center of the lanes, maneuvering at turns and bridges (shown in figure 1).

     Data generated via autonomous mode driving stored in 'run1' and the corresponding generated video file is name as 'run1.mp4'.

3. CNN model

   Before proceeding for the CNN model, images from the simulator (3-images per frame, corresponding to left, right and center, for giving different perspective of track ahead, shown in figure 2) are normalized, processed (by flipping the image horizontally) and cropped (for area of interest). Due to flipping of image, multiply the steering angle by a factor of -1 to get the steering angle for the flipped image.

   NvidiaNet architecture has been used. As an input this model takes cropped images from the input image of the shape (160,320,3). 60 pixels from the top and 25 pixels from the bottom of an image frame is cropped to remove the extra information like trees, sky and dashboard.

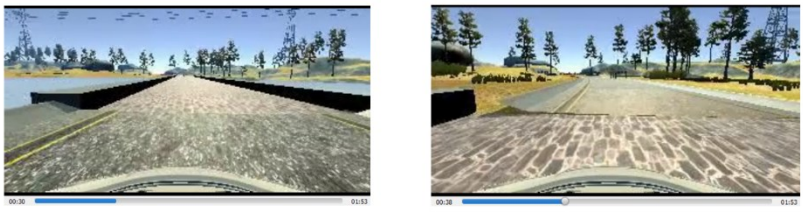

Figure 1: Instances of successful training from the final result: (a) Before the beginning of bridge, (b) transition from bridge to pitch road, (c)(d)(e) Maneuvering turns and avoiding to go out of track

Figure 2: Different perspective from Left, Center and Right Camera

The overall strategy for deriving a model architecture was to predict the steering angle based on the image's (3 images) from camera mounted on the car. As steering angle is associated with 3 camera images, a correction factor (of 0.3) for left and right images are taken, since the steering angle is captured by the center angle. For left images, steering angle is increased by correction factor, while for right images, steering angle is reduced by correction factor.

I have kept 20% of the data for validation set and remaining for training set.

The major issue is steering the vehicle at steep curves, where tweaking on hyperparameters and batch size is required. Although I didn't tried much, once my model start working.

Used model parameter for training:

i. No. of epochs = 12
ii. Optimizer Used - Adam
iii. Validation Data split- 0.30
iv. Generator batch size = 32
v. Correction factor = 0.3

vi. Loss Function Used - MSE
vii. Dropout rate = 0.35

4. As there were lot of instructions provided, which initially make me confused; later I understood my actual work is on the model.py file to generate model.h5 (model3.h5 here), which will be further used drive.py to call.