Behavioral Cloning Project

Required Files:

The submission includes a

model.py file - Containing the script used to create and train the model,

drive.py – To drive the car in the autonomous mode,

model.h5 – Trained convolutional network model,

a writeup report – containing the contents about training and model and

video.mp4 – containing the video of the car running in the autonomous mode.

Quality of code:

Submission includes the functional code, using the Udacity provided simulator and drive.py file, the car can be driven autonomously around the track by using the following command.

```
python drive.py model.h5
```

The code in model.py is usable and contains the appropriate comments to understand the code and can train the neural network and generate the network model file (.h5).

Model Architecture and Training Strategy:

Has an appropriate model architecture been employed for the task?

I have used NVIDIA architecture for the project. I have tried LeNet model and a custom-built model, both of them haven't produced the desired response.

The model has the convolutional layers with depths ranging from 24 to 64. The model has ReLu activation for each convolutional layer and have a lambda layer before the first convolutional layer for normalization.

Has an attempt been made to reduce overfitting of the model?

Dropout layers are added between the convolution layers to reduce the overfitting of the model.

Have the model parameters been tuned appropriately?

I have used Adam optimizer, no necessity for tuning the learning rate. I have tuned the values of Dropout percentage and the steering values with respect to left and right images to get appropriate reduction in training and validation errors.

Is the training data chosen appropriately?

I have used the data of driving the car in both the directions and utilized more training data from driving the car in the turns which require robust maneuvers. This model utilizes the images from the center, left and right cameras. I have manipulated the steering measurement values for right and left images.

Architecture and Training Documentation:

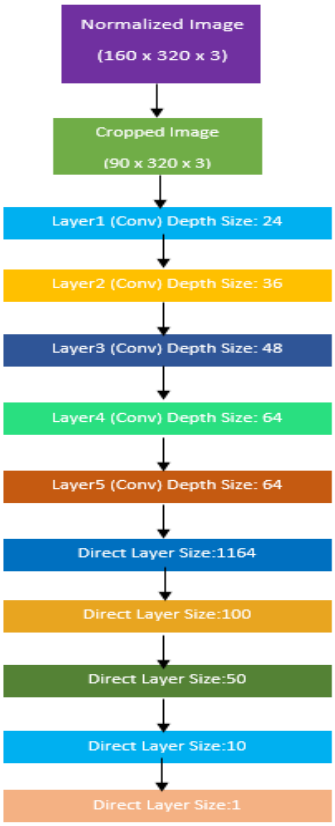
Solution Design Approach:

The strategy of the model is to predict the steering input based on the images from the center, left and right images obtained.

To do that I have used the NVIDIA network architecture to train my model and I have evaluated the performance of the model by using the validation data set (20% of the total – 48000 images) to validate the performance of the model.

I have evaluated the performance of the model by using the model to run the vehicle autonomously in the first track and I have tuned the adjusted steering values for the left and right images to improve the performance. This has resulted in the ability of the car to drive around the track without going out of the road.

Final Model Architecture:



Is the creation of the training dataset and training process documented?

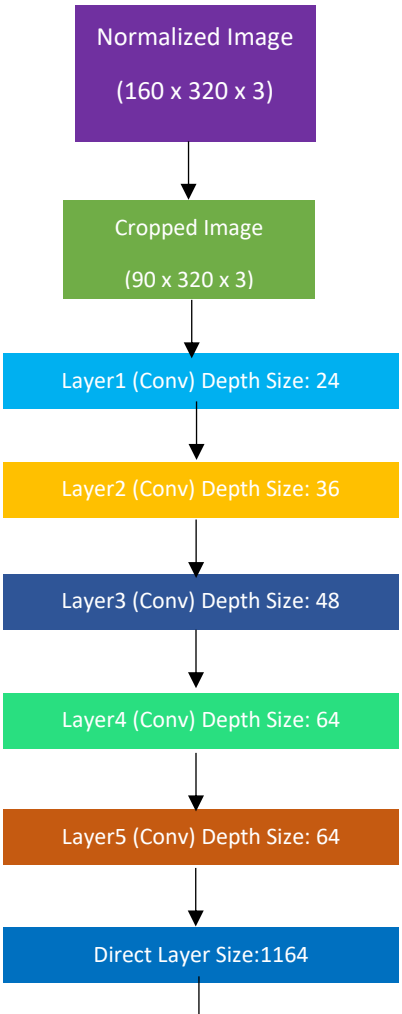
I have done a center lane driving for two laps in the one direction and I have gathered the data. I have also driven it in the other direction to make the neural network generalize for different driving scenarios.

The following image gives an example for the center lane keeping (Center camera) image.

The following image gives an example for the recovery to the center of the lane. The images from left, center and right cameras are shown in the order below.

Then I got the additional training data by driving the car in the opposite direction, instead of flipping the data.

After the collection process, I have split the acquired data into validation and training dataset. 20% of the total data is validation data set. I have then randomly shuffled the data set for the training process and I have validated the training process by tracking the accuracy in the validation data set to avoid the overfitting of data. I have used Adam optimizer so the tuning of learning rate is not necessary.


Normalized Image
(160 x 320 x 3)
Cropped Image
(90 x 320 x 3)
Layer1 (Conv) Depth Size: 24
Layer2 (Conv) Depth Size: 36
Layer3 (Conv) Depth Size: 48
Layer4 (Conv) Depth Size: 64
Layer5 (Conv) Depth Size: 64
Direct Layer Size:1164