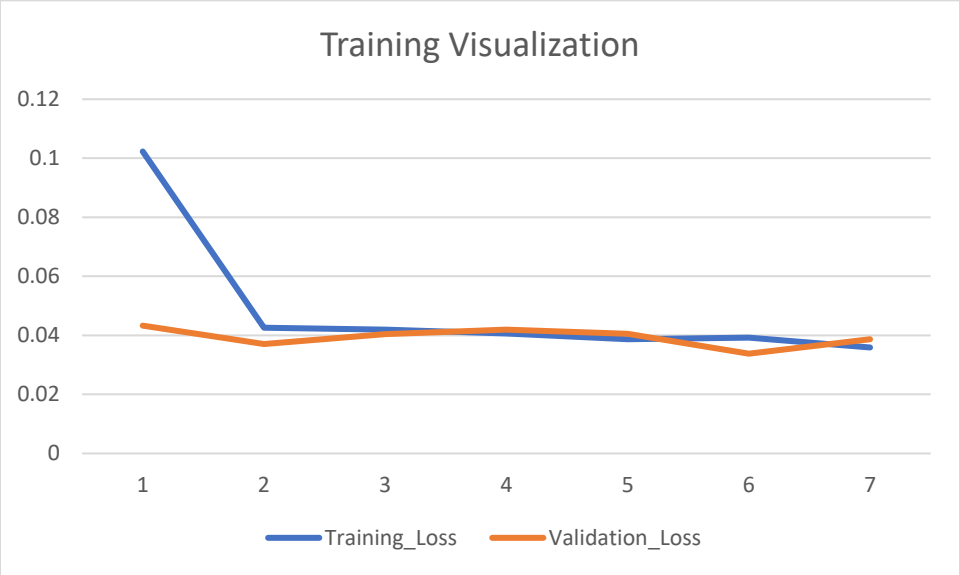
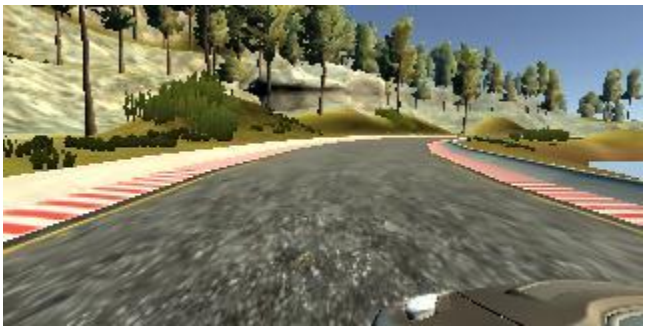
Behavioral Cloning Project

Image References

Model Visualization

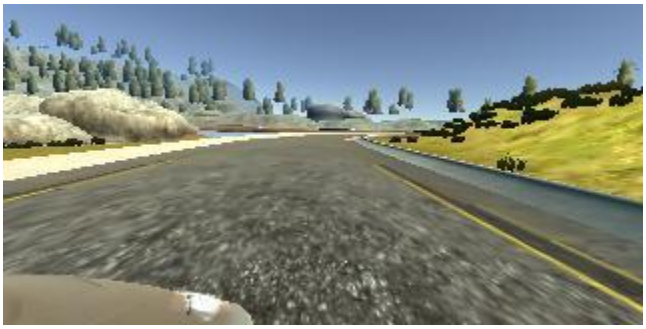


Recovery Image 1 (taken from left camera)



Normal Image

Recovery Image 2 (taken from right camera)



Flipped

Rubric Points

The submission include:

- clone_2.py and clone_3.py (2 and 3 stands for different iteration, with 3 being better than 2)
- drive.py
- model.h5 and model_2.h5 (this corresponds to clone_2.py and clone_3.py respectively).
- Writeup

Code

Drive.py is as it was when it was when the project repo was downloaded. Besides that, files that were used to create final deliverable (video) were:

- clone_3.py: this contains the generator from which data was drawn, as well as the neural network structure with keras.
- model_2.h5: this is the model file in which the trained model is saved.

Here is a summary of the model architecture employed:

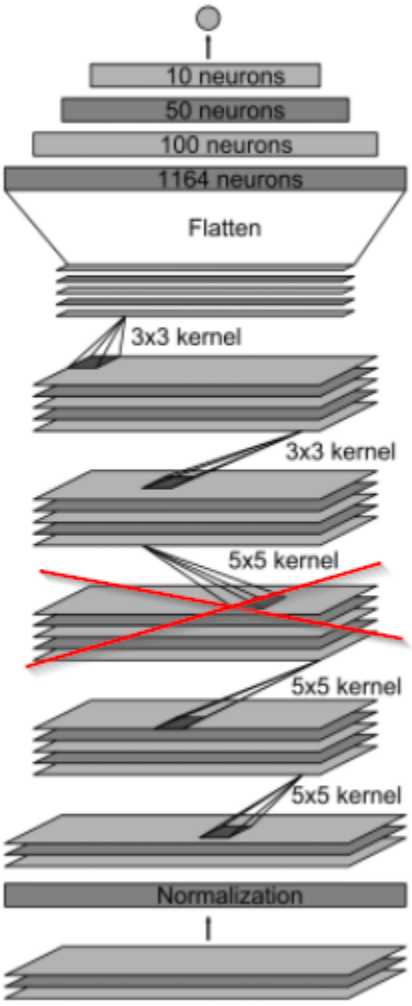
```
#building the model
from keras.models import Sequential
from keras.layers import Flatten, Dense, Lambda
from keras.layers.convolutional import Convolution2D
from keras.layers.pooling import MaxPooling2D
from keras.layers import Cropping2D

model = Sequential()
#mdoel normalization and mean centering using Lambda
model.add(Lambda(lambda x: x/255 - 0.5, input_shape=(160,320,3)))
#adding cropping layers
model.add(Cropping2D(cropping=((70,25),(0,0))))
#adding several convolutional layers
model.add(Convolution2D(24,5,5,subsample=(2,2),activation='relu'))
model.add(Convolution2D(48,5,5,subsample=(2,2),activation='relu'))
model.add(Convolution2D(46,3,3,activation='relu'))
model.add(Convolution2D(64,3,3,activation='relu'))

#flattening output, does not affect batch size
model.add(Flatten())
model.add(Dense(100))
model.add(Dense(50))
model.add(Dense(10))
model.add(Dense(1))
```

Several convolutions were used with relu activation were used to introduce non-linearity into the model, followed by three directly connected layers. And finally one single output in which the steering prediction is given.

In terms of overfitting prevention, the approach taken was reducing the number of epoch from 10 to 7. Dropout layers were not used since the car ran fine in autonomous mode.

The model utilized adam optimizer, so the learning rate was not tuned manually.

The model structure is based on the NVIDIA model shown earlier. However, instead of 3 5x5 convolutional layer, there are instead 2. The reason for this deduction is because the results (at least for application of training steering angle) are mostly identical. Therefore a layer is deleted to save on training time.



The data was created by:

- 2 counter clock wise, smooth laps
- 1 clock wise, smooth laps
- 1 clock wise, recover lap, alternating between left and right side of the road. This is for recovery.

The model saved in the repo, however, was trained using the data already provided by the project repo.

The recovery was trained using the two cameras on the sides, as described in page 12 “Using Multiple Cameras” in the project intro:

```
#create augmented images for the three images
aug_center_image = cv2.flip(center_image,1)
aug_left_image = cv2.flip(left_image,1)
aug_right_image = cv2.flip(right_image,1)
#load in measurements
measurement_center = float(batch_sample[3])
measurement_left = measurement_center + steering_correction
measurement_right = measurement_center - steering_correction
aug_measurement_center = measurement_center*-1.0
aug_measurement_left = measurement_left*-1.0
aug_measurement_right = measurement_right*-1.0
```

Results

With everything set up, the network is first trained using the following parameters:

- Batch_size = 32
- Steering_correction = 0.2
- Epoch = 3

With this set up, the car ran off the tight turn without clear lane marking.

This is perhaps due to the car not being able to correct itself quickly enough. The steering correcting is therefore increased for the next training.

- Batch size = 32
- Steering_correction = 0.5
- Epoch = 7

It is noticed loss increased significantly once the steering_correction was increased. The epoch was in turn increased to counter that.

With this setup, the car swerved left and right more frequently (as one may expect since the correction magnitude is now greater, there tends to be over correction).
The car did not make it pass the bridge.

Subsequently, the steering_correction is then lowered to 0.35 and an epoch of 7.. This yielded good result.

Track 2

Model_2.h5 is also tested on track 2, the result of which is captured in the video “video_2”. Having never seen data from this track, the model performed a lot worse in comparison to track one.

Most notably, the model has a lot of trouble identifying the middle of the road during the darker section of the track:

In addition, the car has a lot of trouble identifying the middle of the road. This is most likely caused by the addition of the divider lane marker.

During the well lit, and longer corners, the model behaved as expected. This is reasonable, given these sections showed great deal of similarity with the data given to the model from track 1.

Note that the video was recorded with numerous disengagement (human intervention). Wherever the video skips frame, it signifies a disengagement.

Given more time, I would retrain the model with data for the second track with transfer learning. In this scenario, the data from the second track can be considered “similar data”. Therefore the case “large data set, similar data” would be applicable. Using a screenshot taken from the “Transfer Learning” segment, the retrain process can be summarized as follows: