Self Driving Car Engineer Nanodegree
Project 3: **Behavioral Cloning**

Mano Paul

The goals of the project was to:
- Use the simulator to collect data of good driving behavior
- Build, a convolution neural network in Keras that predicts steering angles from the images captured from the simulator
- Train and validate the model with a training and validation set
- Test that the model would allow a vehicle to successfully drive around track one full lap without leaving the road

Files Submitted & Code Quality

1. Submission includes all required files and can be used to run the simulator in autonomous mode

My project includes the following files:
- ***model.py*** containing the script to create and train the model
- ***drive.py*** for driving the car in autonomous mode using the trained model
- ***model.h5*** containing a trained convolution neural network
- ***video.mp4*** which is the video recording of the vehicle driving autonomously around the track for atleast one full lap
- ***video.py*** which is the python script to generate the video of the vehicle driving autonomously
- ***Autonomous-Driving-Behavior-Cloning.pdf*** that includes the summary of this project with findings and work effort demonstrated

2. Submission includes functional code

Using the Udacity provided simulator and my drive.py file, the

car can be driven autonomously around the track by executing
`python drive.py model.h5`

3. Submission code is usable and readable

The model.py file contains the code for training and saving the convolution neural network.

Model Architecture and Training Strategy

1. An appropriate model architecture has been employed

The model used here was the NVIDIA model which is shown in the model.py file. It consisted of convolutions neural network (CNN) with the following layers, filter sizes (3x3 and 5x5) and depths (24 to 64) as shown in the section “Final Model Architecture”

It used RELU layers to ensure that it was non-linear and the normalization of the data was accomplished by using the Keras lambda layer before the data was model through the CNN.

2. Attempts to reduce overfitting in the model

In order to reduce overfitting of the data, the model used dropout layers with a keep probability of 0.9 and was trained using various training data captures. The model was tested by running it to the simulator and ensuring that the car stayed on the road.

3. Model parameter tuning

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

4. Appropriate training data

The training data captures was generated by running the car through the tracks 1 and tracks 2 in the simulator. This included a combination of center driving, recovering from left and right, and to avoid the bias to the left, the car was driven in the opposite direction in track 1 and captured into the training samples dataset.

Model Architecture and Training Strategy

1. Solution Design Approach

The overall strategy for deriving a model architecture was to

1. Read the camera images (center, left and right) and the steering angles from the data captured in the training run using the simulator
2. Augment the simulator car dataset
    a. by flipping the images
    b. by flipping the steering angles
3. Split the training data into a training set and a test set (2% of the training set)
   The image and steering angle data was split into a training and validation set.
4. Generate the training and validation dataset with a mini batch size of 32
5. Using the nvidia CNN and the Keras model, my model was trained
   I use a convolution neural network model similar to the nvidia model. I thought this model might be appropriate because according to the NVIDIA paper "*End to End Learning for Self Driving Cars",* this CNN model maps raw pixels from single front-facting camera directly to steering commands with minimum training data from humans.

This will allow a vehicle to be able to operate on local roads without lane markings and on highways as well, functioning well in areas with limited or unclear visual guidance. Additionally this model will allow for automatic learning of internal representations of needed steps to detect useful road features with only the human steering angle as the training signal without having the need to explicitly train the detect outline of roads and signals etc.

6. Combating overfitting
   Then I found out in some runs that the model was overfitting which was evident by the low mean squared error on training set but high mean squared error on the validation test.

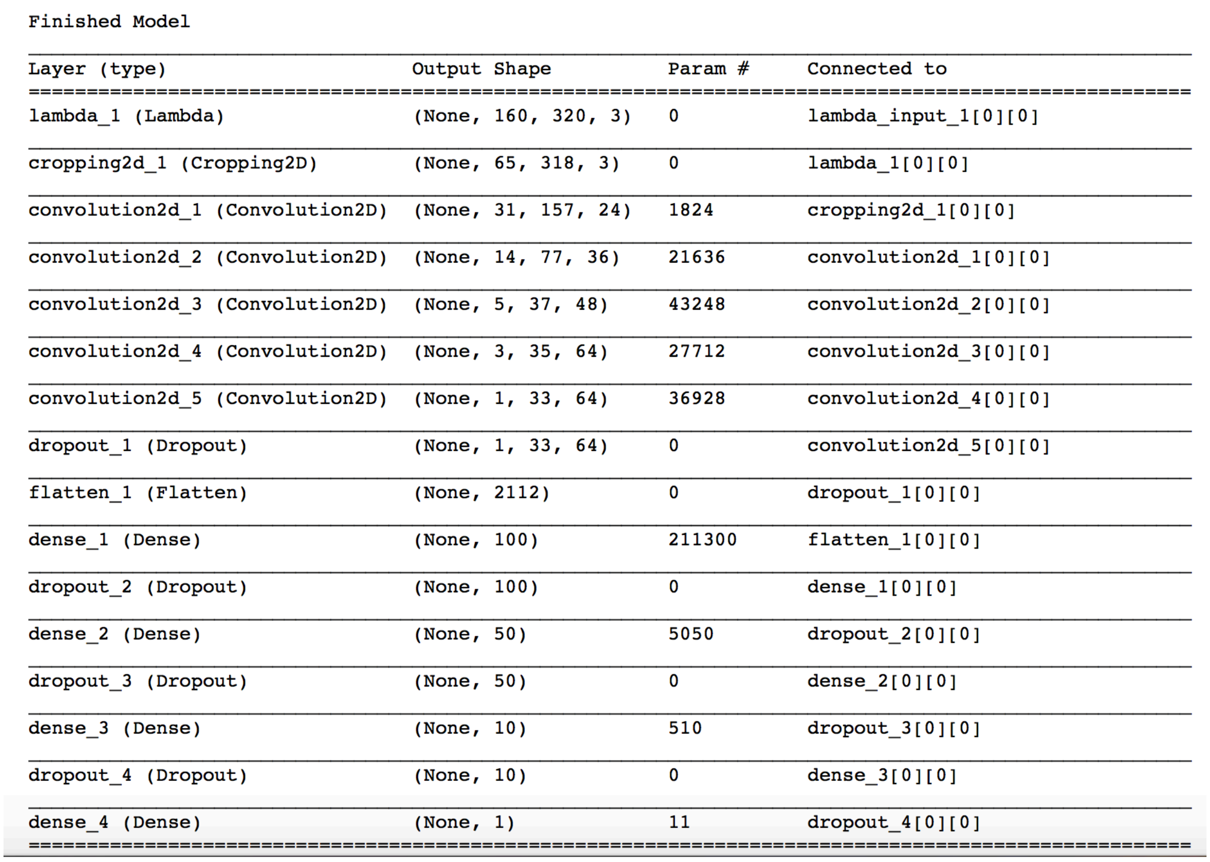
   To combat the overfitting, I modified the model by reducing the number of epochs from 5 to 2 and adding dropouts so that mean squared error on the validation set was low. The number of epochs was 2 as higher number of epochs kept making the training overfit.

7. The final step was to run the simulator to see how well the car was driving around track one. There were a few spots where the vehicle fell off the track. To improve the driving behavior in these cases, I had to
   a. change steering angle correction factor
   b. change the keep probability ratio for dropouts
   c. change batch and sample size
   d. change

In order to gauge how well the model was working, I ran the model over 100 times atleast. At the end of the process, the vehicle is able to drive autonomously around the track without leaving the road.

2. Final Model Architecture

The final model architecture consisted of a convolution neural network with the following layers and layer sizes as shown below.

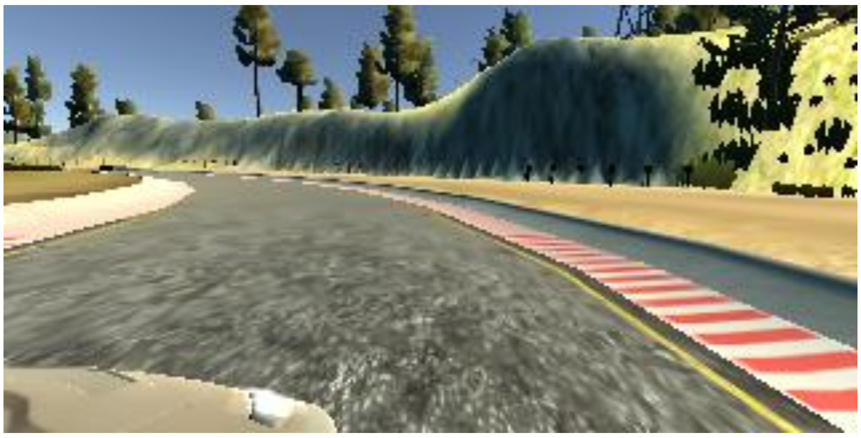


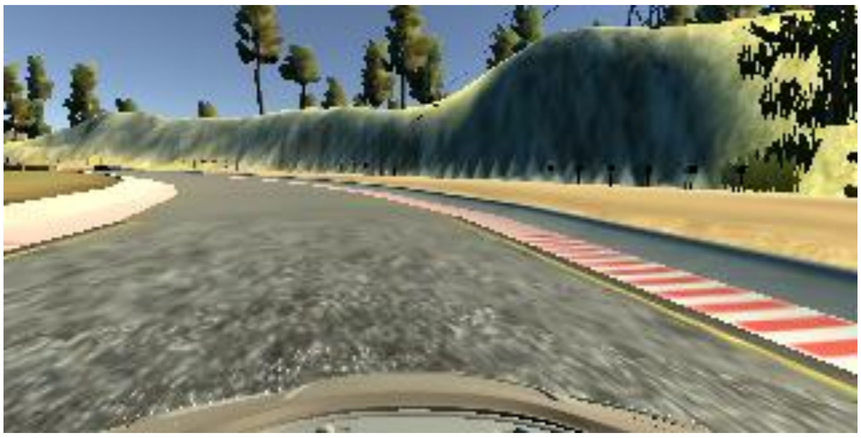
```
Finished Model

____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
lambda_1 (Lambda)                (None, 160, 320, 3)   0           lambda_input_1[0][0]
____________________________________________________________________________________________________
cropping2d_1 (Cropping2D)        (None, 65, 318, 3)    0           lambda_1[0][0]
____________________________________________________________________________________________________
convolution2d_1 (Convolution2D)  (None, 31, 157, 24)   1824        cropping2d_1[0][0]
____________________________________________________________________________________________________
convolution2d_2 (Convolution2D)  (None, 14, 77, 36)    21636       convolution2d_1[0][0]
____________________________________________________________________________________________________
convolution2d_3 (Convolution2D)  (None, 5, 37, 48)     43248       convolution2d_2[0][0]
____________________________________________________________________________________________________
convolution2d_4 (Convolution2D)  (None, 3, 35, 64)     27712       convolution2d_3[0][0]
____________________________________________________________________________________________________
convolution2d_5 (Convolution2D)  (None, 1, 33, 64)     36928       convolution2d_4[0][0]
____________________________________________________________________________________________________
dropout_1 (Dropout)              (None, 1, 33, 64)     0           convolution2d_5[0][0]
____________________________________________________________________________________________________
flatten_1 (Flatten)              (None, 2112)          0           dropout_1[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 100)           211300      flatten_1[0][0]
____________________________________________________________________________________________________
dropout_2 (Dropout)              (None, 100)           0           dense_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 50)            5050        dropout_2[0][0]
____________________________________________________________________________________________________
dropout_3 (Dropout)              (None, 50)            0           dense_2[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 10)            510         dropout_3[0][0]
____________________________________________________________________________________________________
dropout_4 (Dropout)              (None, 10)            0           dense_3[0][0]
____________________________________________________________________________________________________
dense_4 (Dense)                  (None, 1)             11          dropout_4[0][0]
====================================================================================================
```

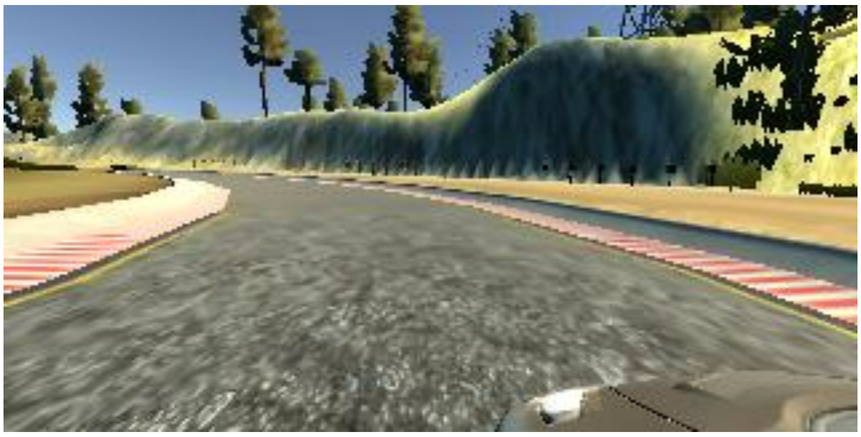
3. Creation of the Training Set & Training Process

To capture good driving behavior, I first recorded two laps on track one using center lane driving. Here is an example image of center lane driving:

I then recorded the vehicle recovering from the left side and right sides of the road back to center so that the vehicle would learn to predict steering angles to center itself.
These images show what a recovery looks like starting from ... :

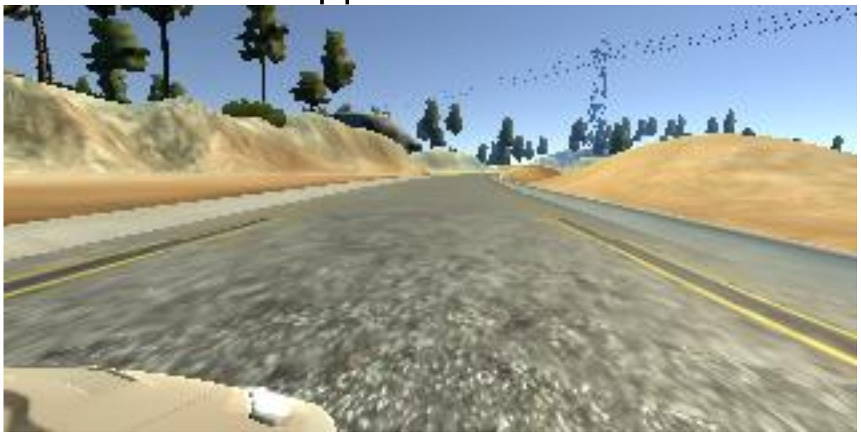


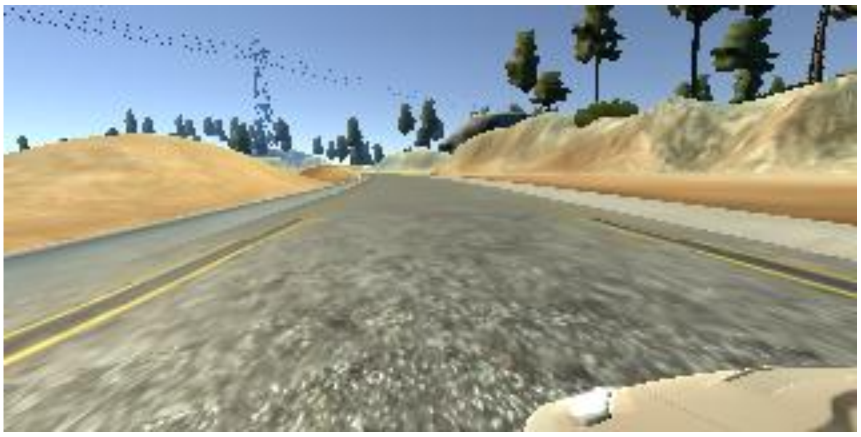




Then I repeated this process by reversing the direction and driving the car around to get more data points.

To augment the data sat, I also flipped images and angles thinking that this would ... For example, here is an image that has then been flipped:





After the collection process, I then preprocessed this data by flipping the images horizontally and flipping the steering angle. I attempted to crop the images and to augment the dataset but ran into issues with the neural network processing on incompatible array sizes and so resorted to using the cropping function in Keras to do the cropping while training instead of preprocessing. I would have preferred the cropping to be a preprocessing steps as it would have reduced training time.

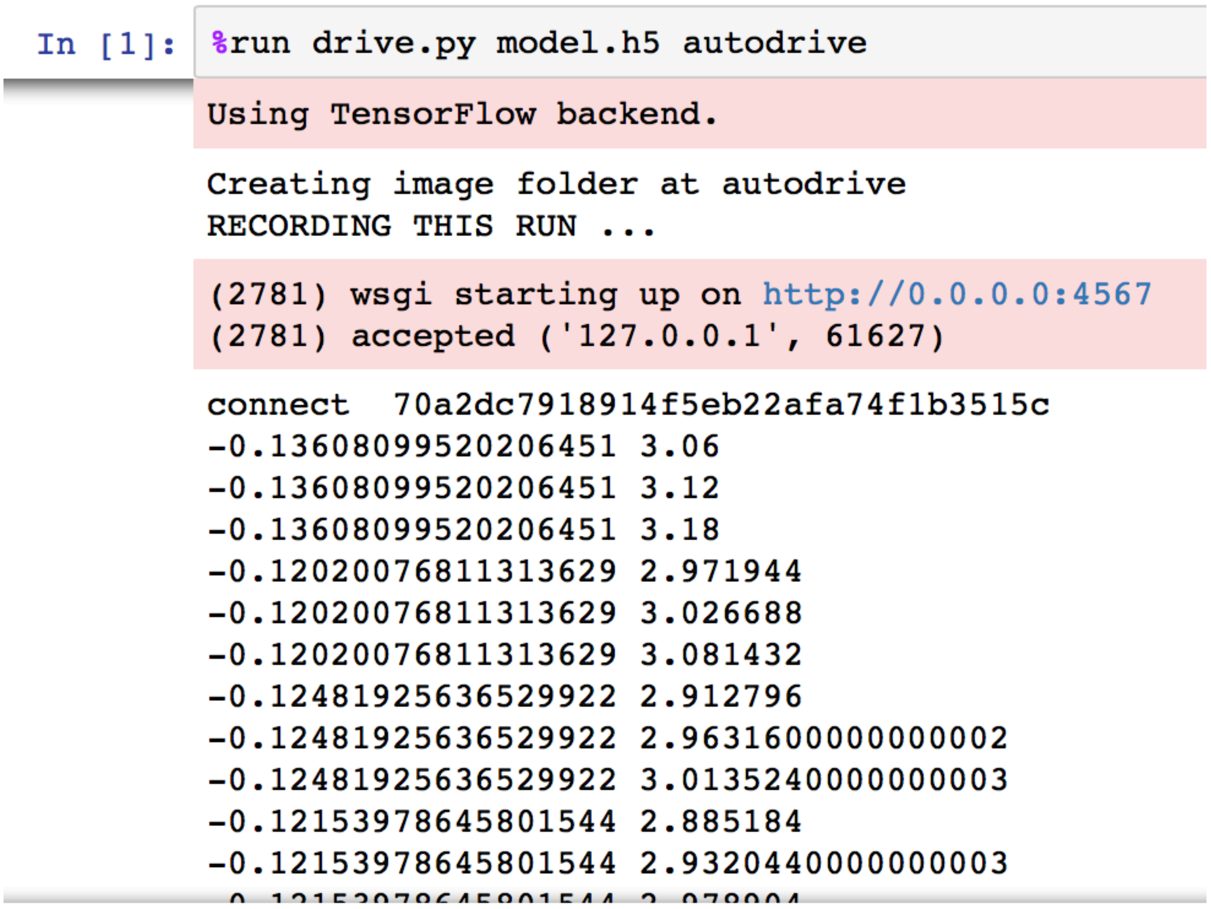
I finally randomly shuffled the data set and put 0.2 of the data into a validation set.

I used this training data for training the model. The validation set helped determine if the model was over or under fitting.

The ideal number of epochs was 2.

I used an adam optimizer so that manually training the learning rate wasn't necessary.

Using the drive script (drive.py) and the trained Keras model (model.py), I tested to see if the vehicle will stay on the road in an autonomous mode by predicting the steering angles automatically. This created the autodrive folder where the images from the simulation run was saved



```
In [1]: %run drive.py model.h5 autodrive

Using TensorFlow backend.

Creating image folder at autodrive
RECORDING THIS RUN ...

(2781) wsgi starting up on http://0.0.0.0:4567
(2781) accepted ('127.0.0.1', 61627)

connect  70a2dc7918914f5eb22afa74f1b3515c
-0.13608099520206451 3.06
-0.13608099520206451 3.12
-0.13608099520206451 3.18
-0.12020076811313629 2.971944
-0.12020076811313629 3.026688
-0.12020076811313629 3.081432
-0.12481925636529922 2.912796
-0.12481925636529922 2.9631600000000002
-0.12481925636529922 3.0135240000000003
-0.12153978645801544 2.885184
-0.12153978645801544 2.9320440000000003
```

Then using the video generation script (video.py), the video.mp4 was generated using the images from the simulation run in the autodrive folder.

```
In [2]: %run video.py autodrive --fps 48
```

```
Creating video autodrive.mp4, FPS=48
[MoviePy] >>>> Building video autodrive.mp4
[MoviePy] Writing video autodrive.mp4
```

```
100%|██████████| 3502/3502 [00:10<00:00, 326.48it/s]
```

```
[MoviePy] Done.
[MoviePy] >>>> Video ready: autodrive.mp4
```