C. D. Vijay
Udacity Car ND Program

Report P3: Behavioral Cloning

Table of Contents

1. Model Architecture

NVIDIA architecture was employed with several modifications

A modified NVIDIA architecture [1] was employed and tailored to meet more modest memory and computation resources available.

This network contains five convolution layers [Convolution2D()] with a kernel sizes of 5x5 and 3x3 with depth ranging from 24-64 (p34nvmod53.py, lines 114-126), interspersed with five MaxPooling layers [MaxPooling2D()]. Each convolutional layer has an activation component (RELU). The network is flattened followed by three [Fully Connected] Dense () layers (ibid., lines 130-134).

DATASET

The dataset was created by manually driving the car in the simulator for 2 laps in both directions and supplemented with recovery data (coming back on road after going off of it).

Two solutions were tried: Each is documented in its own source code file (See Files submitted, files 1 and 6). The main differences between files 1 and 6 are:

File 1 reads only the center image for now along with its steering angle.

Data is augmented by flipping the image vertically and horizontally to remove any turn bias, along with reversing the steering angle (line 85-86).

File 6 (lines 30-70) uses a generator routine to generate mini-batches of data. In addition, it reads all the three image files (center, left and right), and adjusts the center steering angle by 0.15 for its left and right counterparts. Only the left and right images are flipped horizontally and vertically.

So, a total of five images result from every entry in the driving log. A batch size of 16 (set to a small # to overcome memory problems on the PC) yields a total of 80 images along with 80 steering angles, which are funneled via arrays to the model.

Data is normalized in the model using a Keras Lambda layer and cropped to remove the extraneous information from the bottom and top of the images.

2. Training Strategy

Final Model Architecture

The final model architecture consisted of a convolution neural network with the following layers, kernel sizes and depths:

1. Convolutional layer (with RELU) – 5x5 – Depth: 24 followed by Max pooling layer

2. Convolutional layer (with RELU) – 5x5 – Depth: 36 followed by Max pooling layer

3. Convolutional layer (with RELU) – 5x5 – Depth: 48 followed by Max pooling layer

4. Convolutional layer (with RELU) – 3x3 – Depth: 64 followed by Max pooling layer

5. Convolutional layer (with RELU) – 3x3 – Depth: 64 followed by Max pooling layer

6. Three (3) fully connected layers

The network configuration, as mentioned earlier, is based on the seemingly simple yet elegant architecture NVIDIA has used [1]. This model was the final one chosen after many iterations and combinations, as no combination of convolution layers, kernel sizes, depths, dropouts and pooling were producing satisfactory results, i.e., the self-driving car would eventually veer off the track.

Attempts to use Dropout layers resulted in lower accuracy in this network configuration, no matter when inserted before and/or after flattening, and were thus dropped!

The model was tested by running it through the simulator and ensuring that the vehicle stayed on track. The output video stays on track for about a lap length. Further tuning using the new generator routine is in progress for better results.

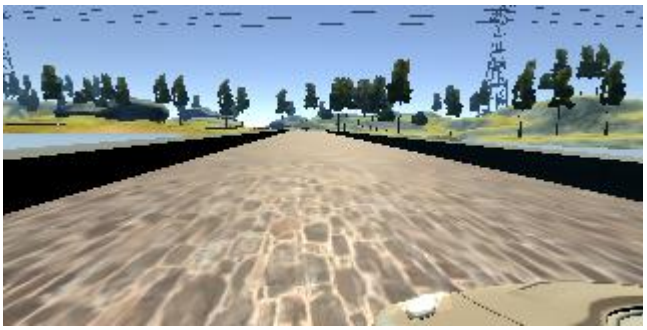
As an “Adam Optimizer” was used (File 1, line 143), the learning rate was not tuned. Batch size and # of epochs, however, were.

The Keras model is shared by the two source files (no difference).

3. Generator, Track and Simulation Images

The training data was chosen to train the network, and split into training / validation samples (80/20).

TRACK 1 IMAGES



GENERATOR (File 6 only, p35gen2.py)

A generator was used to generate “samples-on-the-fly” as memory problems were faced even on the GPU. But, for some reason, the training time was far too slow even on the GPU!

Some of the other parameters were:

Batch size: 32

Number of epochs: 3

Training / Validation sample split: 80/20 % of total samples

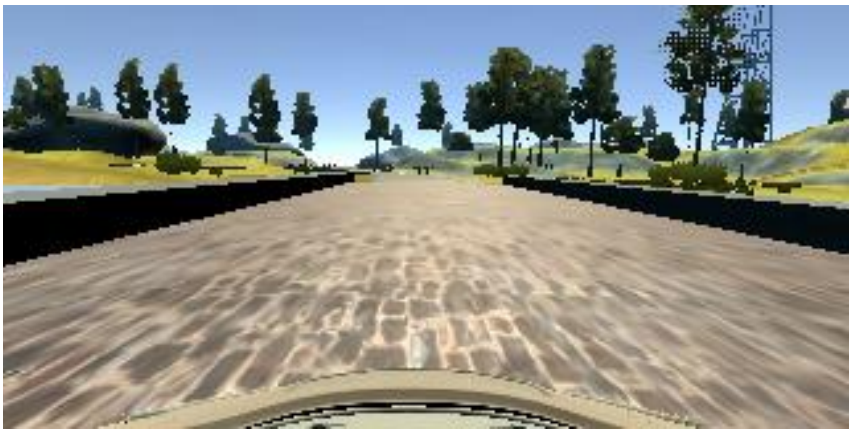
```
def generator(samples, batch_size=32):
    num_samples = len(samples)
    while 1: # Loop forever so the generator never terminates
        #shuffle(samples)
        for offset in range(0, num_samples, batch_size):
            batch_samples = samples[offset:offset+batch_size]

            images = []
            angles = []
            for batch_sample in batch_samples:
                fname = batch_sample[0].split('\\')[-1]
                name = './IMG/'+ fname
                center_image = mpimg.imread(name)
                center_angle = float(batch_sample[3])
                images.append(center_image)
                angles.append(center_angle)

            # trim image to only see section with road
            X_train = np.array(images)
            y_train = np.array(angles)
            yield sklearn.utils.shuffle(X_train, y_train)
```

Matplotlib was used to read the images (not OpenCV) to prevent any mixup in color spaces as reportedly simulator data outputs RGB images, and OpenCV reads in BGR space.

IMAGE CROPPING

To remove the extraneous information from the images during training and to make computation faster, Keras’ Cropping2D was used. As the setting of (70, 20) for the Cropping2D () always produced a strange error, the image was trimmed 30 pixels and 20 pixels from the top and bottom respectively.

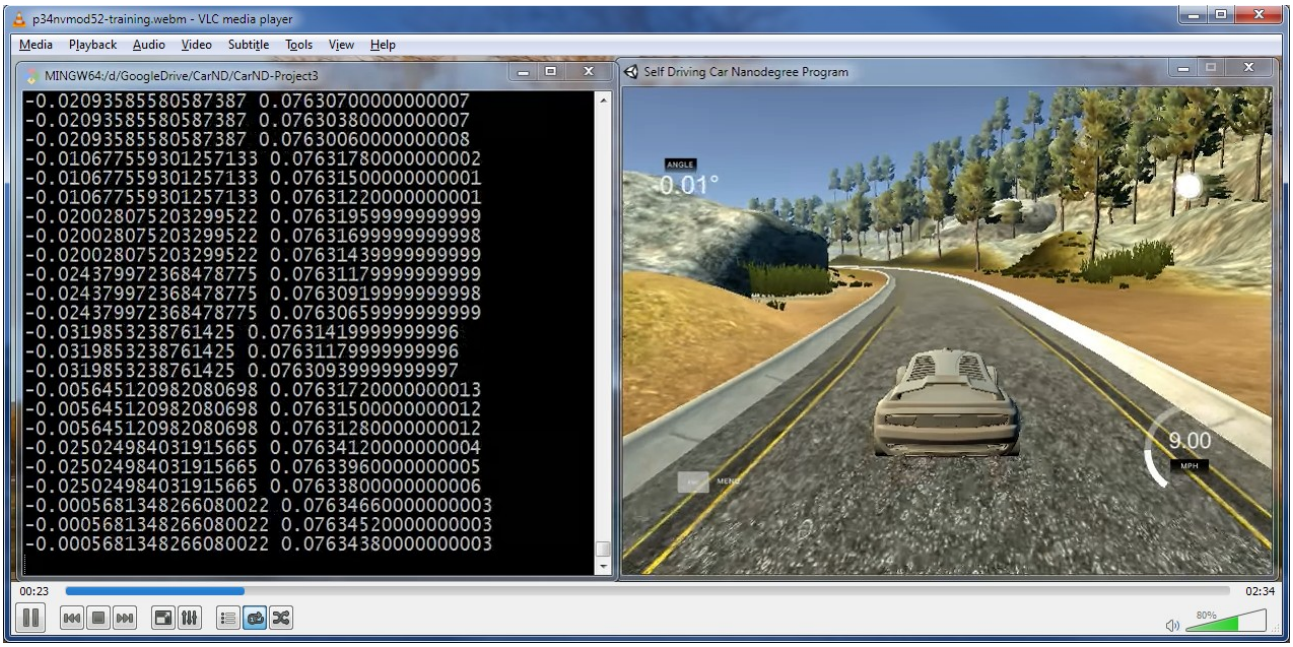
Original Image (320x160)



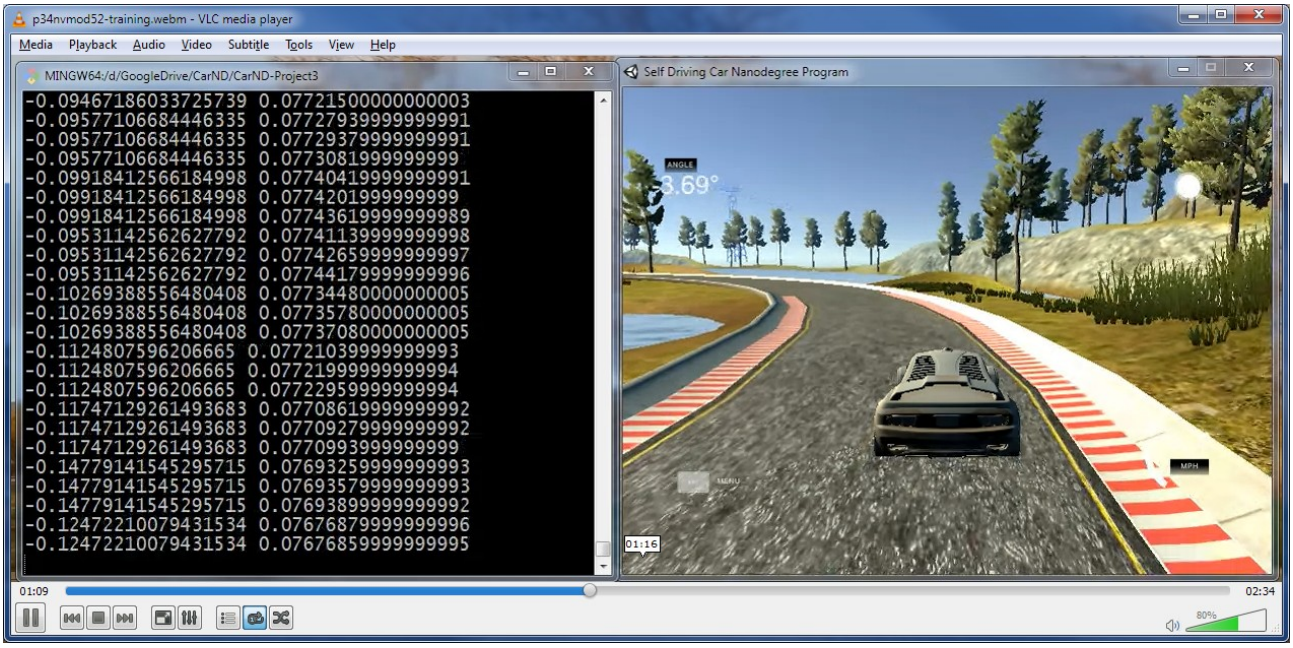
Cropped image (320x110)

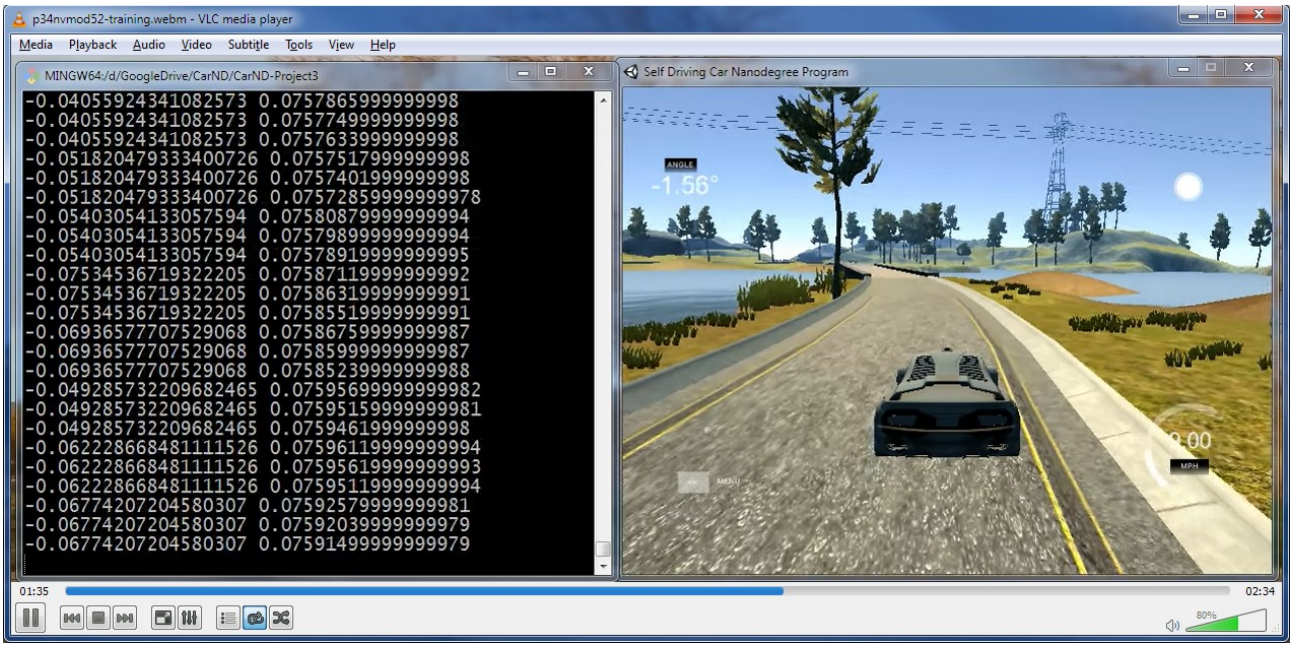
Training takes a long time, even on a recent 4GB GPU! The validation accuracy never rose above ~72%.

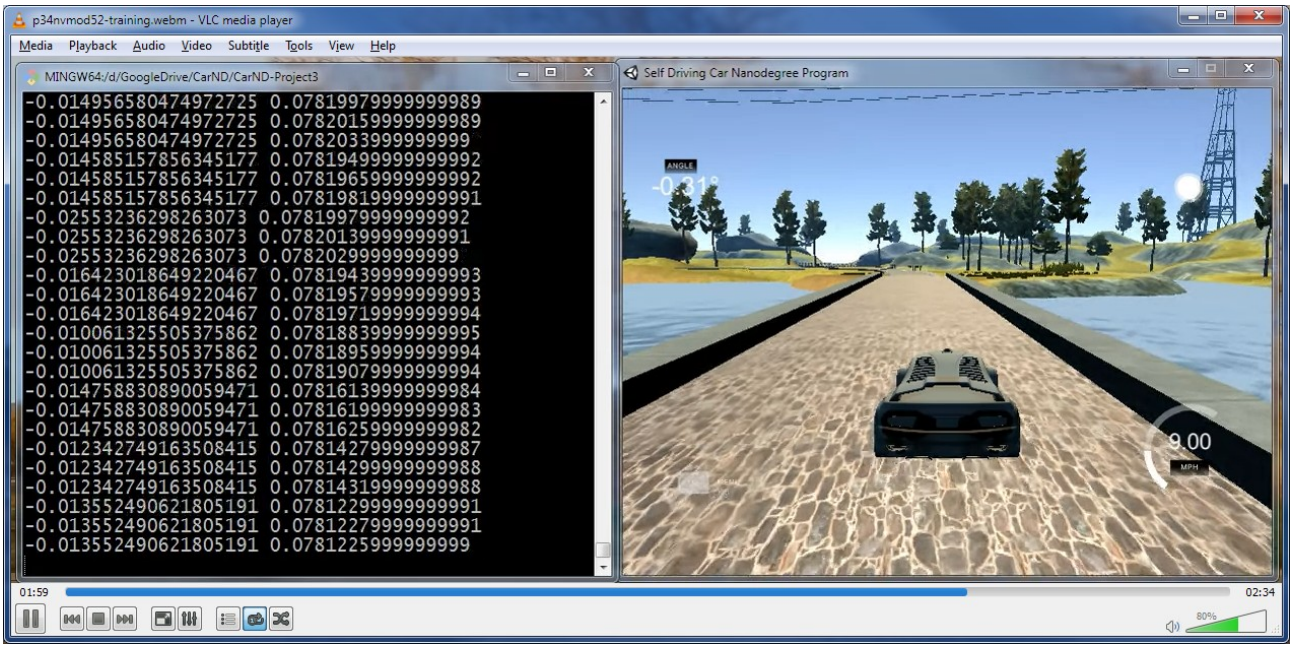
Screenshots of the Simulation

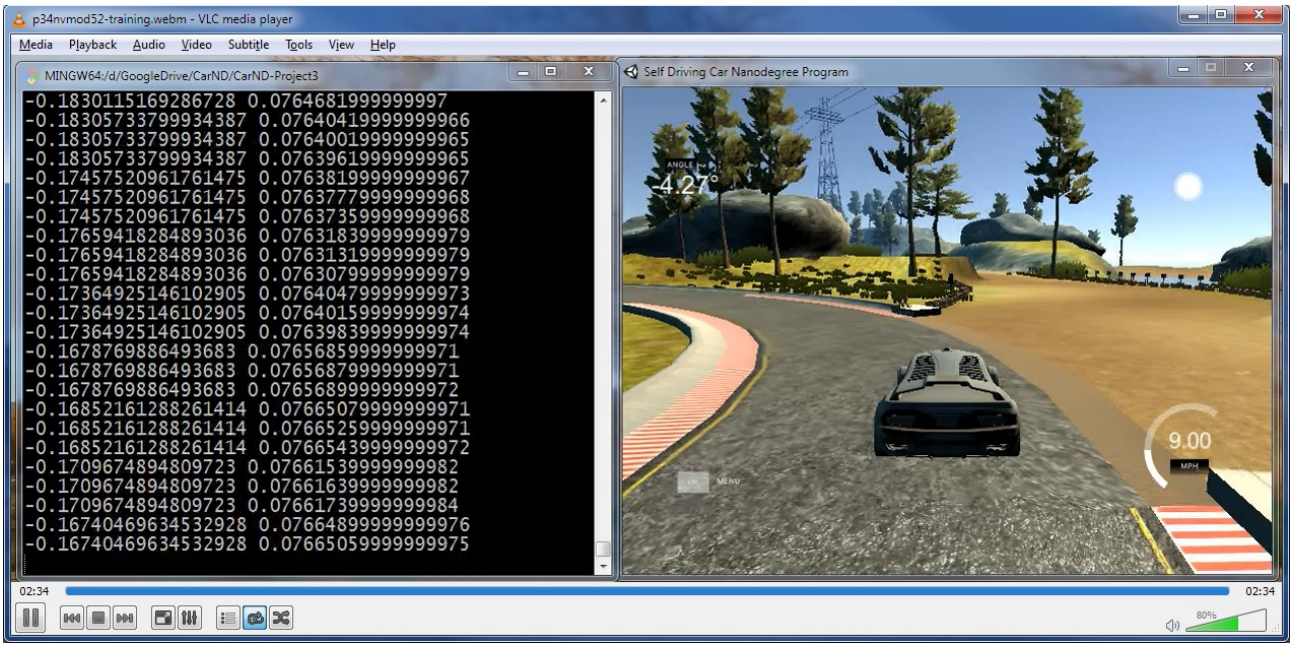


The final step was to run the simulator to see how well the car drove around Track 1. As can be seen from the images next (and the video link on next page), the vehicle is able to drive autonomously around the track without leaving it.




p34nvmod52-training.webm - VLC media player
Media Playback Audio Video Subtitle Tools View Help
MINGW64:/d/GoogleDrive/CarND/CarND-Project3
-0.04055924341082573 0.0757865999999998
-0.04055924341082573 0.0757749999999998
-0.04055924341082573 0.0757633999999998
-0.051820479333400726 0.0757517999999998
-0.051820479333400726 0.0757401999999998
-0.051820479333400726 0.07572859999999978
-0.05403054133057594 0.07580879999999994
-0.05403054133057594 0.07579899999999994
-0.05403054133057594 0.07578919999999995
-0.07534536719322205 0.07587119999999992
-0.07534536719322205 0.07586319999999991
-0.07534536719322205 0.07585519999999991
-0.06936577707529068 0.07586759999999987
-0.06936577707529068 0.07585999999999987
-0.06936577707529068 0.07585239999999988
-0.049285732209682465 0.07595699999999982
-0.049285732209682465 0.07595159999999981
-0.049285732209682465 0.0759461999999998
-0.062228668481111526 0.07596119999999994
-0.062228668481111526 0.07595619999999993
-0.062228668481111526 0.07595119999999994
-0.06774207204580307 0.0759257999999981
-0.06774207204580307 0.07592039999999979
-0.06774207204580307 0.07591499999999979
Self Driving Car Nanodegree Program
ANGLE
-1.56°
9.00
MPH
01:35
02:34
80%


p34nvmod52-training.webm - VLC media player
Media Playback Audio Video Subtitle Tools View Help
MINGW64:/d/GoogleDrive/CarND/CarND-Project3
-0.014956580474972725 0.07819979999999989
-0.014956580474972725 0.07820159999999989
-0.014956580474972725 0.0782033999999999
-0.014585157856345177 0.07819499999999992
-0.014585157856345177 0.07819659999999992
-0.014585157856345177 0.07819819999999991
-0.02553236298263073 0.0781997999999992
-0.02553236298263073 0.07820139999999991
-0.02553236298263073 0.0782029999999999
-0.016423018649220467 0.07819439999999993
-0.016423018649220467 0.07819579999999993
-0.016423018649220467 0.07819719999999994
-0.010061325505375862 0.07818839999999995
-0.010061325505375862 0.07818959999999994
-0.010061325505375862 0.07819079999999994
-0.014758830890059471 0.07816139999999984
-0.014758830890059471 0.07816199999999983
-0.014758830890059471 0.07816259999999982
-0.012342749163508415 0.07814279999999987
-0.012342749163508415 0.07814299999999988
-0.012342749163508415 0.07814319999999988
-0.013552490621805191 0.07812299999999991
-0.013552490621805191 0.07812279999999991
-0.013552490621805191 0.0781225999999999
Self Driving Car Nanodegree Program
ANGLE
-0.31°
9.00
MPH
01:59
02:34
80%



5. Files Submitted

Files in the submission include

Link to the *project code*

1. P34nvmod53.py

2. drive.py

3. P34nvmod53.h5

4. P31-Report.pdf (this report file)

5. *Simulation Video on Youtube.com*

6. p35gen2.mod

To execute autonomous driving simulation

```
Prompt:\> python drive.py p34nvmod53.h5
```

The Udacity Simulator should be started first. (Default Windows Desktop 64-bit.exe).

REFERENCE

1. arXiv:1604.07316v1 [cs.CV] 25 Apr 2016

["End to End Learning for Self-Driving Cars", Mariusz Bojarski, et al, NVIDIA Corp., Holmdel, NJ 07735]

2. Udacity CarND Program Course Material