Behavioral Cloning Report

Thomas Tian

November 13, 2017

1 Project Introduction

In this project, I used what I've learned about deep neural networks and convolutional neural networks to clone driving behavior. I trained, validated and tested a model using Keras. The model outputs a steering angle to an autonomous vehicle.

I used a simulator provided by Udacity where I can steer a car around a track for data collection. I used image data and steering angles to train a neural network and then used this model to drive the car autonomously around the track.

2 Data Collection and Pre-Processing

The training data set was obtained by manually operating an vehicle in the simulator provided by the Udaciy. Three images were captured when the vehicle runs on the track at each time index. Figure 1, Figure 2 and Figure 3 show the three view images captured by the cameras. Note that all three images have the same associated steer angle, how ever, I added an angle offset, $\delta_S = 0.2^o$ and $\delta_S = -0.2^o$ to the left and right camera view, respectively, to generate more data. Moreover, the all the images are flipped and the corresponding steering angle is multiplied by -1 to generate more data. Therefore, the number of the total training images can be calculated by

$$N = 2 * n, \tag{1}$$

where n is the number of the images captured. To generalize the training model and make the vehicle have the ability to steer back when it deviates from the center of the track, the data I obtained by manually driving the vehicle is augmented with training set provided by Udacity to make the training more effective.

The image I got from the camera has more than enough information I need to predict the steering angle, such as the front of the vehicle, sky. Therefore, all the images are cropped (70 pixels from the top and 20 pixels from the bottom) to eliminate the extra information. Moreover, all the images's brightness level (HSV) in the data set are randomly adjusted to generate more data.

Figure 1: Center camera view.

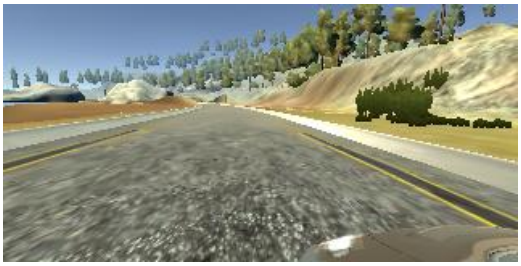

Figure 2: Left camera view.

Figure 3: Right camera view.

3 Model Architecture and Training Strategy

3.1 Model Architecture

A deep neural network was employed to train the model. Figure 5 shows the structure of the used CNN. The model I used is a modified version from the NVIDIA CNN model. The NVIDIA has proven to be a excellent model for image classification and prediction, thus I used this model as a start point. In practice, I added two additional fully connected followed by dropout to prevent over fitting, which improved the accuracy a little.

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, 320, 3)	0	lambda_1[0][0]
convolution2d_1 (Convolution2D)	(None, 31, 158, 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, 128)	73856	convolution2d_4[0][0]
flatten_1 (Flatten)	(None, 4224)	0	convolution2d_5[0][0]
dense_1 (Dense)	(None, 400)	1690000	flatten_1[0][0]
dropout_1 (Dropout)	(None, 400)	0	dense_1[0][0]
dense_2 (Dense)	(None, 200)	80200	dropout_1[0][0]
dropout_2 (Dropout)	(None, 200)	0	dense_2[0][0]
dense_3 (Dense)	(None, 100)	20100	dropout_2[0][0]
dense_4 (Dense)	(None, 75)	7575	dense_3[0][0]
dense_5 (Dense)	(None, 50)	3800	dense_4[0][0]
dense_6 (Dense)	(None, 25)	1275	dense_5[0][0]
dense_7 (Dense)	(None, 20)	520	dense_6[0][0]
dense_8 (Dense)	(None, 1)	21	dense_7[0][0]

Total params: 1,971,767
Trainable params: 1,971,767
Non-trainable params: 0

Figure 4: CNN model.

3.2 Training Strategy

The collected images were shuffled and split into two sets, training set and validation set. The training set contains 80% of the collected data and the validation set contains 20% of the collected data. Since the amount of the training data is significantly larger compared with the training data in the traffic sign classifier project, thus a data generator was implemented to generate data as the training process needed. The training uses an Adam optimizer so that the learning rate is determined by the optimizer. Four training Epochs were used until the training was completed.

4 Training Result

I trained the model with different camera views and noticed that the accuracy using only center camera images for training, 60%, is much larger than the accuracy using images from three cameras, 30%. However, when testing the trained model in the simulator, the model trained with all three cameras' images showed better performance as the vehicle has the ability to steer back when it deviates from the center. The demovideo.mp4 shows the vehicle running on the track using trained model.

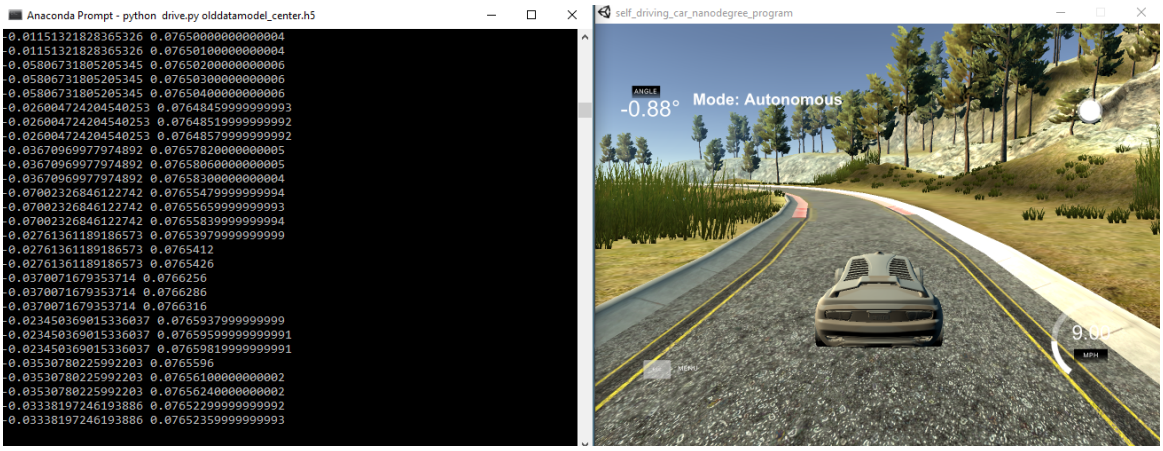

Figure 5: Vehicle running in autonomous mode.