Behavioral Cloning for Self-Driving Cars

The goal of this project is to train a deep neural network to clone driving behavior by using images to predict steering angles. Once the model is trained, it will be used to drive a car autonomously around a test track in Udacity's driving simulator.

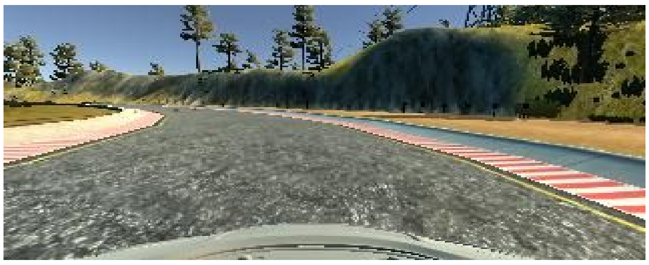
Introduction

The objective of this project is to apply deep learning principles to effectively teach a car to drive autonomously in a simulated driving application. The simulator includes both training and autonomous modes, and two tracks on which the car can be driven.

As a starting point, my model was inspired by Nvidia's architecture from their white paper End to End Learning for Self-Driving Cars. However I have written my own model and got satisfactory output from simulator.

In training mode, user generated driving data is collected in the form of simulated car dashboard camera images and control data (steering angle, throttle, brake, speed). Using the Keras deep learning framework, a convolutional neural network (CNN) model is produced using the collected driving data (see `model.py`)

Sample image



Sample line from driving_log.csv

center_2017_06_19_19_34_59_012.jpg,left_2017_06_19_19_34_59_012.jpg,right_2017_06_19_19_34_59_012.jpg, 0, 0, 0, 7.695253

Approach

Data Collection

One of the most important part of this project is to collect training data. It took me a while to get training data from the simulator in “training” mode. Two laps from the center of the track and one-one lap each from each of the road from left and right side was enough to get started with below mentioned model.

Data Augmentation

Cv2.flip was used to flip image from training data and likewise steering angle measurement was multiplied by -1.

Cleaning Dataset

80% of training data with steering angle measurement of 0.0 was removed from training set as noise reduction.

All the images are preprocessed into HLS color space.

Model Architecture

Layer	Description
Input	160 x 320 x 3
Cropping2D	crop_shape= ((80,20),(1,1))
Normalize	Lambda x:x/255.0 - 0.5
Convolution2d	5 x 5 kernels, filters = 6, activation = 'relu'
MaxPooling2D	`pool_size=(2, 2), strides=None, border_mode='valid'`
Convolution2d	5 x 5 kernels, filters = 6, activation = 'relu'
MaxPooling2D	`pool_size=(2, 2), strides=None, border_mode='valid'`
Convolution2d	5 x 5 kernels, filters = 6, activation = 'relu'
MaxPooling2D	`pool_size=(2, 2), strides=None, border_mode='valid'`
Dropout	5 % , to address overfitting
Flatten	
Dense	Fully connected NN layer (120)
Dense	Fully connected NN layer (84)
Dense	Fully connected NN layer (1)

Above model is trained using following configs for optimization :
loss='mse',optimizer='adam'

With above model we get following accuracy :

Training loss: 0.0320
Validation loss: 0.1017

Conclusion and Discussion

It is very important in this project to collect training data carefully. I am yet to explore data provided by udacity . Making changes to the model rarely seemed to have quite the impact that a change to the fundamental makeup of the training data typically had.

To manage my time effectively I chose to conclude my efforts as soon as the model performed satisfactorily on the track. I fully plan to revisit this project when time permits.

There is lot of scope of improvements such as using left and right images as discussed in lecture series. I am very happy with the outcome of this project and I look forward to one day testing some of my models on a real car.