Behavioral Cloning

Behavioral Cloning Project

The goals / steps of this project are the following:

- Use the simulator to collect data of good driving behavior
- Build, a convolution neural network in Keras that predicts steering angles from images
- Train and validate the model with a training and validation set
- Test that the model successfully drives around track one without leaving the road
- Summarize the results with a written report

Rubric Points

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

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
- model.h5 containing a trained convolution neural network
- writeup_report.md or writeup_report.pdf summarizing the results

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.json

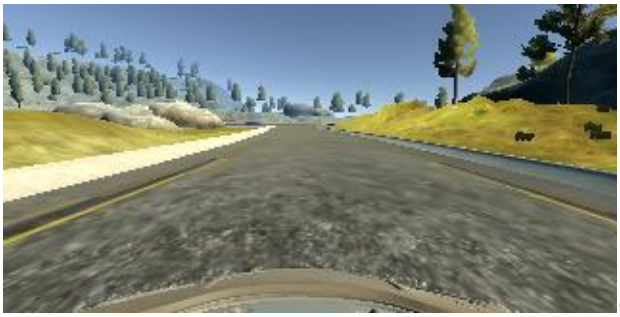
3. Submission code is usable and readable

The model.py file contains the code for training and saving the convolution neural network. The file shows the pipeline I used for training and validating the model, and it contains comments to explain how the code works.

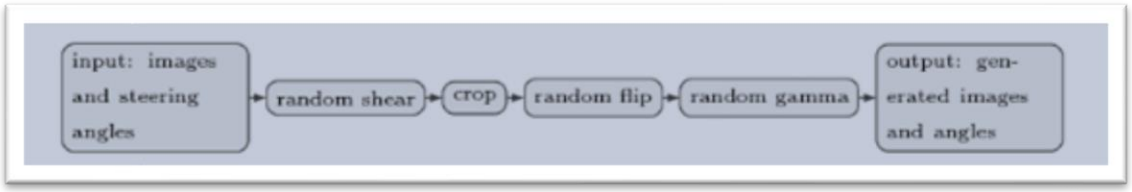
Model Architecture and Training Strategy

 **right**

 **left**

 **center**

The dataset consists of 24108 images (8036 X 3). The training track contains a lot of shallow turns and straight road segments. Hence, most of the recorded steering angles are zeros. Therefore, preprocessing images and respective steering angles are necessary to generalize the training model for unseen tracks such as our validation track.

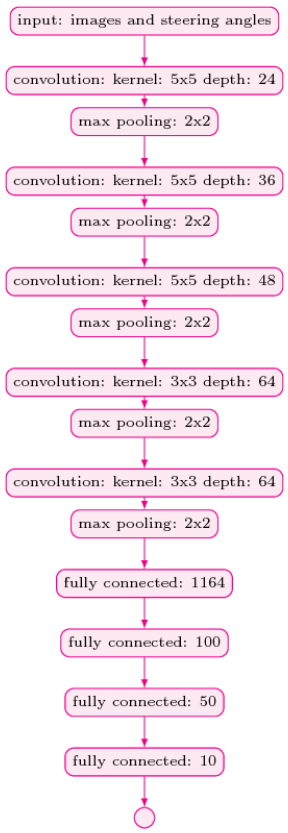


The dataset was randomly sheared, cropped, flipped, gamma corrected and resized to preprocess the data.

Network Architecture

1. Solution Design Approach

This convolutional neural network architecture was inspired by NVIDIA's End to End Learning for Self-Driving Cars paper. The main difference between this model and the NVIDIA mode is that this uses MaxPooling layers just after each Convolutional Layer to cut down training time.



Training

Even after cropping and resizing training images (with all augmented images), training dataset was very large, and it could not fit into the main memory. Hence, fit generator API of the Keras library was used for training the model.

We created two generators namely:

- train_gen = helper.generate_next_batch()
- validation_gen = helper.generate_next_batch()

Batch size of both train_gen and validation_gen was 64. We used 20032 images per training epoch. It is to be noted that these images are generated on the fly using the document

processing pipeline described above. In addition to that, we used 6400 images (also generated on the fly) for validation. We used Adam optimizer with 1e-4 learning rate. Finally, when it comes to the number of training epochs we tried several possibilities such as 5, 8, 10, 25 and 50. However, 8 works well on both training and validation tracks.

Results:

The results are included in the video provided. The car successfully navigates through the track autonomously.