Behavioral Cloning

Writeup Template

You can use this file as a template for your writeup if you want to submit it as a markdown file, but feel free to use some other method and submit a pdf if you prefer.

Behavioral Cloning Project

Overview

Project aims to drive the Vehicle in the specific lane lines with the use of Deep learning convolutional neural networks. Image data is generated with the use of Lane driving car simulator provided by udacity.

The main components involved in Behavioral cloning project involves:

1. *Preprocessing Image data.*
2. *Segmenting the useful data and Normalization of data.*
3. *Augmentation with the flipping images, translation and adding shadow.*
4. *Data generator.*
5. Simulator in driving the car.

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
  - Preprocessing2.py
  - Model2.py
- drive.py for driving the car in autonomous mode
- model.h5 containing a trained convolution neural network
  - model_Pdata.h5
  - model_Pdata.json
- 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.h5
```

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.

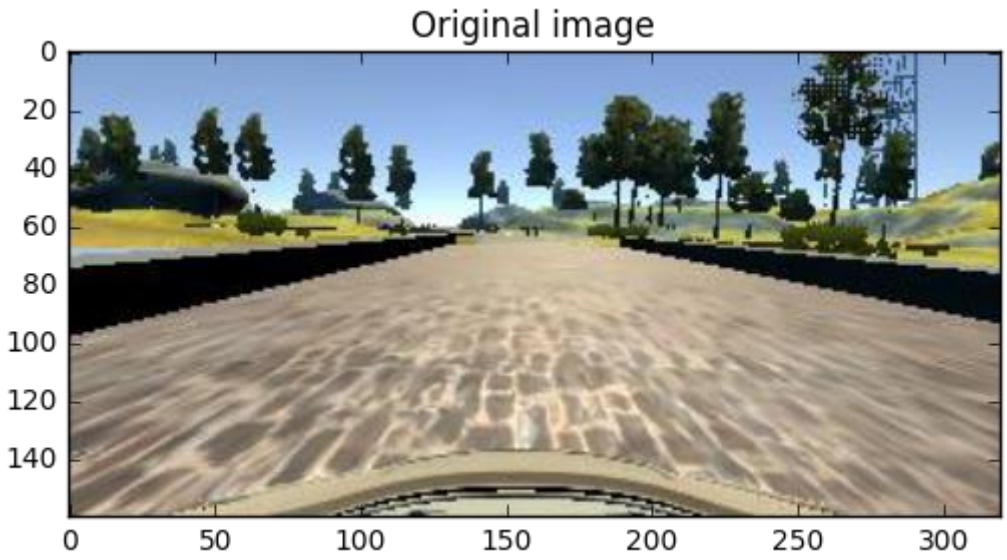
Model Architecture and Training Strategy

Approach

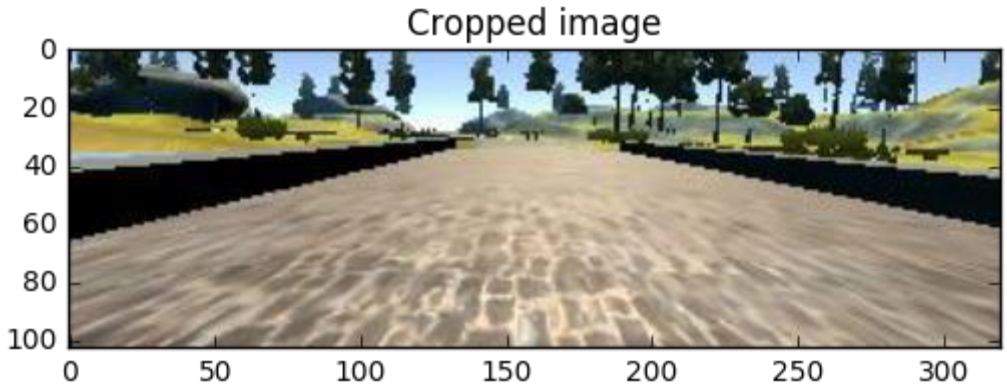
The training data which is generated through udacity simulator includes original data of complete Front view camera. Original data of 160x320 is included with trees and other unwanted information to model in training, to reduce the time computation complexity and to get better results, image cropped with 70 pixels from top and 25 pixels from down .

The following image is an example of the results.

Original Image



After removing sky and hood

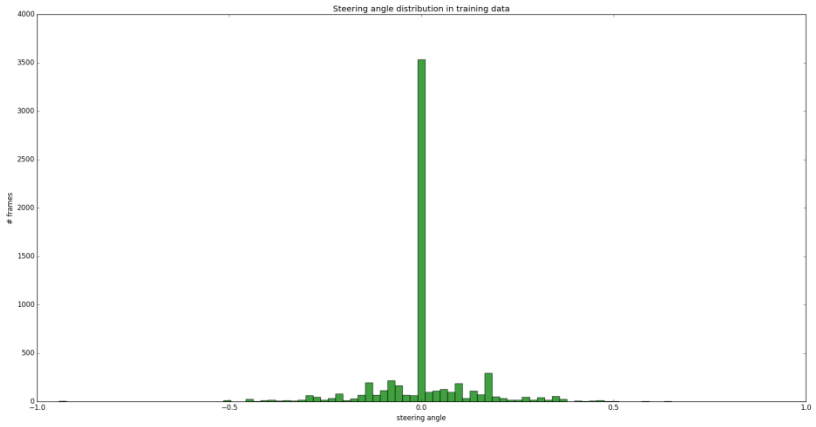


2. Attempts to reduce overfitting in the model

The model contains dropout layers in order to reduce overfitting (model.py lines 21).

The model was trained and validated on different data sets to ensure that the model was not overfitting (code line 10-16). The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track.

- First Thing from my model is, It mainly faced high bias, It was not at all learning the features even after several attempts of increasing the Network size, Augmenting the data, but still network could not learn.
- After several attempts, I have observed high skew data,



After that I have reduced skew data by removing most of the images which are having 0 degrees steering angle.

Preprocessing

```
Drivinglog = pd.read_csv(load_dir + 'driving_log.csv')
Drivinglog =
pd.concat([Drivinglog[Drivinglog.steering.ne(0)],Drivinglog[Drivinglog.steering.betw
een(-0.01,0.01)][:600]]).reset_index(drop=True)
```

3. Model parameter tuning

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

Here I have attached the Model summary that I have used.

I have concentrated mainly in fine tuning batch size image data to feed and the image augmentation techniques in improving the model.

(model.py line 25).

I

4. Appropriate training data

Training data was chosen to keep the vehicle driving on the road. I used a combination of center lane driving, recovering from the left and right sides of the road .

For details about how I created the training data, see the next section.

For training data I have tried to use the car simulator in generating data, I could not generate as much data with specific speed and other requirements, so I have downloaded the generated data from the specified link in Udacity Student hub center. [https://study-hall.udacity.com/rooms/community:nd013:233565-project-237/community:thread-11298324738-752225?contextType=room]

After consideration of 8000 training image data also I have faced high bias problem in deep learning neural network model. So to make the model to have good precision, I have used data augmentation techniques

Data Augmentation techniques used

- Flipping image data
- Translation of images
- And adding shadow .

Along with the manual data augmentation techniques , I have used keras image generator in increasing the training data.

```
train_datagen = ImageDataGenerator(featurewise_center=False, samplewise_center=False,
featurewise_std_normalization=False,

                                   samplewise_std_normalization=False, zca_whitening=False, zca_epsilon=1e-6,
rotation_range=10, width_shift_range=0.2,

                                   height_shift_range=0.2,shear_range=0, zoom_range=0.1, channel_shift_range=0.,
fill_mode='nearest', cval=0.,

                                   horizontal_flip=False,vertical_flip=False, rescale=None, preprocessing_function=None,
data_format="channels_last")
```

Model Architecture and Training Strategy

1. Solution Design Approach

The overall strategy for deriving a model architecture was to reduce the mean square error of each epoch, which enables model to train well in detecting accurate steering angle .

My first step was to use convolution neural network model similar to the le net model, which have built in previous assignment, but model filters followed could'nt learn the lane line features data.

In order to gauge how well the model was working, I split my image and steering angle data into a training and validation set. I found that my first model had a low mean squared error on the training set but a high mean squared error on the validation set. This implied that the model was overfitting.

To combat the overfitting, I modified the model so that , slowly model was trying to learn the lane curvature features at the end of network layers, I have built a model, which is sequentially reducing the width and size of layers followed by incremental of filters in learning features as max as possible.

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, mainly at the first steep turn, the model is not able to give sufficient steering angle to turn the car in to the specific lane at steep turns, then I have tried to increase the turn angle from training data manually , which is not a feasible approach to solve.

- a = Drivinglog[Drivinglog.columns[3]].astype(float)
- y_data = (a + ((((~a.between(-0.05,0.01)) & (a.gt(0.01))).astype(int) * 0.05) + (((~a.between(-0.05,0.01)) & (a.lt(-0.01))).astype(int) * -0.05))).values[:,None]

to improve the driving behavior in these cases, I have finally found that the data is skewed towards 0 degrees angle as the images data of 0 degrees is much higher, so have filtered maximum 0 degrees images data from main data.

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 (model.py lines 18-24) consisted of a convolution neural network with the following layers and layer sizes ...

Here is a visualization of the architecture (note: visualizing the architecture is optional according to the project rubric)

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
cropping2d_1 (Cropping2D)    (None, 65, 320, 3)        0
_________________________________________________________________
lambda_1 (Lambda)            (None, 65, 320, 3)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 31, 158, 6)        456
_________________________________________________________________
activation_1 (Activation)    (None, 31, 158, 6)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 77, 16)        2416
_________________________________________________________________
```

activation_2 (Activation) (None, 14, 77, 16) 0

_____

conv2d_3 (Conv2D) (None, 5, 37, 32) 12832

_____

activation_3 (Activation) (None, 5, 37, 32) 0

_____

conv2d_4 (Conv2D) (None, 3, 35, 32) 9248

_____

activation_4 (Activation) (None, 3, 35, 32) 0

_____

conv2d_5 (Conv2D) (None, 1, 33, 32) 9248

_____

activation_5 (Activation) (None, 1, 33, 32) 0

_____

flatten_1 (Flatten) (None, 1056) 0

_____

dense_1 (Dense) (None, 1164) 1230348

_____

activation_6 (Activation) (None, 1164) 0

_____

dense_2 (Dense) (None, 100) 116500

_____

activation_7 (Activation) (None, 100) 0

_____

dense_3 (Dense) (None, 50) 5050

_____

activation_8 (Activation) (None, 50) 0

_____

dense_4 (Dense) (None, 10) 510

_____

activation_9 (Activation) (None, 10) 0

_____

dense_5 (Dense) (None, 1) 11

==========

Total params: 1,386,619

Trainable params: 1,386,619

Non-trainable params: 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 follow the track along with the lane lines. These images show what a recovery looks like starting from ... :

Then I repeated this process on track two in order 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:

Image Preprocessing to read each image in a directory and making them as a generalized array to feed Neural Network.

Preprocessing of images included with 5 steps

1. Reading all training images and the driving log csv to create Train features and predicting variable data.

2. Translating the images with the use Random flipping and assigning a steering angle negatively for each flipped image.

3. Randomly adding shadow to each image

4. Concatenating all augmented data

5. Splitting the data to train, test and valid randomlyAfter the collection process, I had X number of data points.

Link to open Simulated video

https://github.com/Nagakiran1/CarND-Behavioral-Cloning-P3/blob/master/run1.mp4