UDACITY: SELF DRIVING NANODEGREE

Project 3: Behaviour Cloning
Prepared By: Rakesh Paul
02/06/2018

Project Goal:

The Goals/Steps of the 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

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
- behavior_cloning.pdf summarizing the results
- video.mp4 of the autonomous driving in Track 1
- video2.mp4 of the autonomous driving in Track 2
- model2.h5 containing a trained convolution neural network

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.

UDACITY: SELF DRIVING NANODEGREE

Model Architecture and Training Strategy

1. Appropriate model architecture has been employed:

The model used consists of 5 convolution layer with each layer followed by 'relu' activations for non-linearity(lines 13- 17 of model.py). The output is then flattened and fed to 4 fully connected layer for getting the final output(lines 19- 25 of model.py). Input images are preprocessed to have normalized value using keras supported lambda layer. Moreover the inputs image are cropped to discard top portion of image till horizon (lines 10- 11 of model.py).

2. Reduce overfitting:

The model was trained and validated on different data sets to ensure that the model was not overfitting.
Dropout layer is added in order to reduce overfitting (line 18 model.py).

3. Model parameter tuning:

The model uses adam optimizer instead of tuning learning rate manually.

4. Training data quality:

Training data was chosen to induce the desired behavior of vehicle on road:

- Driving at the center of road.
- Driving in opposite direction of the track
- Recovery from left and right side of road.

Model Architecture and Training Strategy

1. Solution Design Approach

My first step was to use a convolution neural network model similar to the LeNet architecture. The car went out of track. Later the inputs were normalized and also cropped. This helped the car to move further but could not complete the first turn. Later the nVidia group model was used for processing. The preprocessing techniques were again used before applying the nVidia model. Only a fully connected layer with output 1 was introduced for getting desired output to compare.

In order to gauge how well the model was working, I split my image and steering angle data into a training and validation set(75% training/25% validation). Moreover the images are flipped and augmented with original ones to have more input data to train network. In addition, the left and right camera images are also added along with small correction on the corresponding steering angle values for the image.

I used a dropout layer to prevent the network from overfitting.
The final step was to run the simulator to see how well the car was driving around track one. 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 consisted of 5 convolution neural network with the 4 fully connected layers and layer resolutions are as follows:

Layer	Output
Lambda	(None,160,320,3)
Cropping2D	(None,65,320,3)
Convolution2D	(None,31,158,24)
Convolution2D	(None,14,77,36)
Convolution2D	(None,5,37,48)
Convolution2D	(None,3,35,64)
Convolution2D	(None,1,33,64)
Dropout	(None,1,33,64)
Flatten	(None, 2112)
Dense	(None, 100)
Dense	(None, 50)
Dense	(None, 10)
Dense	(None, 1)

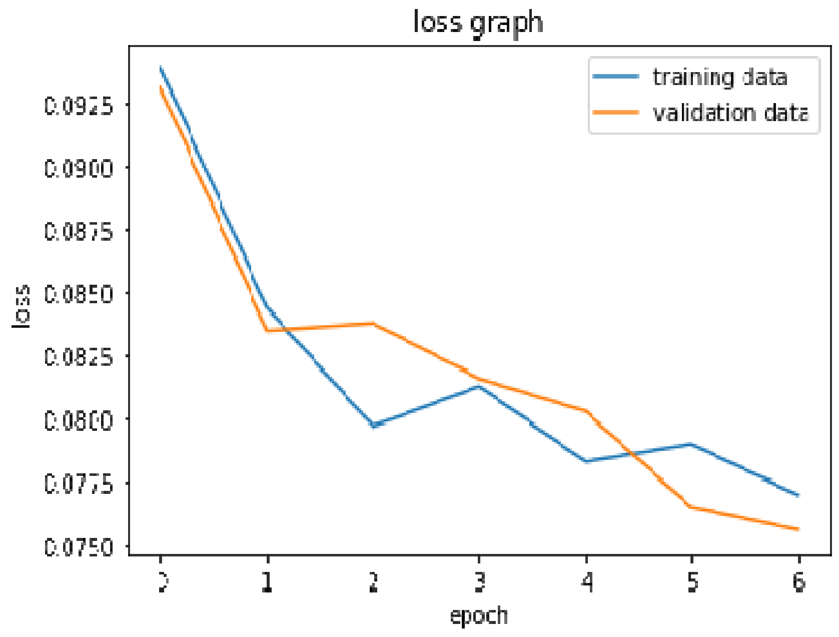
A visualization of input images are provided in **visualize.ipynb**.

3. Creation of the Training Set & Training Process

The training set consists of the following performed on track 1:

- Forward driving keeping vehicle to the center
- Driving in opposite direction.
- Driving to the left and right side of the road and immediately getting back towards the center of road.

The training process was performed for 7 epochs. The data were shuffled randomly. Generator code was used with batch size 128.



After this, the car was driving down the road as expected with getting out of road.