Behavioural Cloning Report

February 12, 2019

0.0.1 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 mode * drive.py for driving the car in autonomous mode * model.h5 containing a trained convolution neural network

0.0.2 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
```

0.0.3 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.

1 Model Architecture and Training Strategy

1.0.1 1. Solution Design Approach

Initially I just wrote a simple model with just to check if things are working. I found some issues in my system which I resolved.

After, that I started digging more with convolution and went through some papers from NVIDIA and STANFORD.

And tried different archictecture like: I tried `RESNET50` pretrained on Imagenet with `15 frozen Resnet`layers and also added some extra fully connected layers but I was not getting the accuracy I wanted.

So, I droped that model and started with `Nvidia` architecture but still it did not worked properly.

So, I thought of `augmenting` the data and taken the images from `left and right cameras` as well and this time it worked far better, it stayed between the lane with litlle flucatuation of steering wheel.

So, I added `BatchNormalization` and to reduce overfitting I added `Dropout` and it worked perfectly well.

alt text

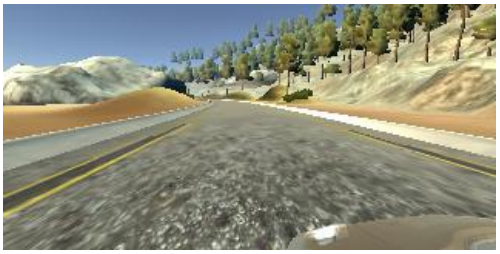
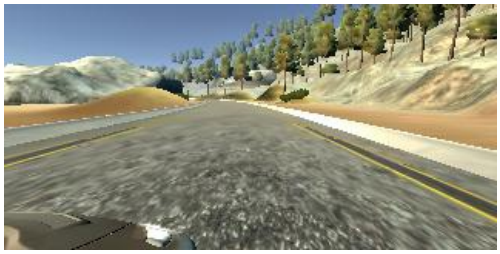
1.0.2 2. Final Model Architecture

layer Name	Explain
Lamda	Normalizaion
Cropping2D	cropping coordinate = ((70,25),(0,0))
Conv2D filters=(5, 5)	strides=(2, 2) activation="relu"
BatchNormalization	
Conv2D filters=(5, 5)	strides=(2, 2) activation="relu"
BatchNormalization	
Conv2D filters=(5, 5)	strides=(2, 2) activation="relu"
BatchNormalization	
Conv2D filters=(3, 3)	strides=(1, 1) activation="relu"
BatchNormalization	
Conv2D filters=(3, 3)	strides=(1, 1) activation="relu"
BatchNormalization	
Flatten	
Dense	size=1164, activation='relu'
Dense	size=100
Dense	size=50
Dense	size=10
Dropout	probability=0.5
Dense	size=1

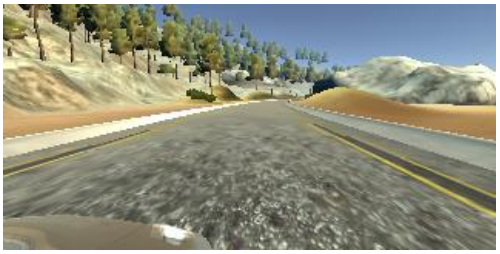
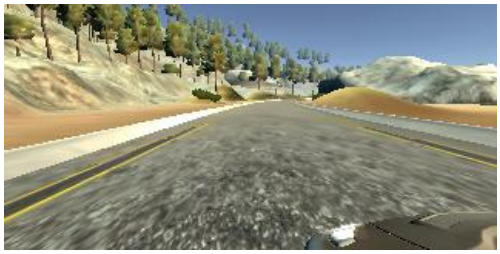
1.0.3 3. Creation of the Training Set & Training Process

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:

Here is the Center Camera Image which I used initially but vehicle was not driving properly.

LEFT IMAGE	RIGHT IMAGE
	

So, I added the images of left and right cameras as well.

FLIPPED LEFT IMAGE	FLIPPED RIGHT IMAGE
	

To make the car drive perfectly I then flipped all images(left, center, right) and here you can see the flipped image of both left and right camera which I have shown above.

1.1 I drove around 2 laps to make the car drive properly

1.2 I have used 80 % of data as Training data and 20 % as validation data

1.3 I have used `generator` for passing the data as now for improving the model even if I add lot of data, memory will be enough and there will be less load on GPU also

1.4 I have used `Dropout` to reduce overfitting and `adam optimizer` so that manually training the learning rate wasn't necessary.