Behavioral Cloning

Writeup Template

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

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:
- Akhil_P3.ipynb 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.pdf summarizing the results
- run1.mp4 file showing run of the first track

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 Akhil_P3.ipynb 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

1.An appropriate model architecture has been employed

My model consists of 5 convolution neural network layers and 4 fully connected layers. Filter sizes for each layer were 5x5, 5x5, 5x5, 3x3 and 3x3 respectively. Depths used for each layer were 24,36,48,64 and 64. The model includes RELU activation functions to introduce nonlinearity. The data is normalized in the model using a Keras lambda layer by dividing each pixel by 255 and then subtracting 0.5. Since top portion of the image is a distraction to train the model I cropped off to 70 pixels. Also I cropped off lower 25 pixels which shows vehicle front.

2. Attempts to reduce over-fitting in the model

To prevent over-fitting model was trained and validated on different data sets. Images were collected by running the vehicle 2 laps counter clockwise direction and also taking images of recovery of vehicle from side-lanes. But this always gave negative steering angles in the output. So I flipped the images and added them to test set with positive steering angles. The model was run only for 3 epochs since running more than that increased validation loss. In the end, the model was tested by running it through the simulator and ensuring that the vehicle could stay on the track.

3. Model parameter tuning

The model used an Adam optimizer, so the learning rate was not tuned manually. Along with that I tuned steering angles when the camera would drive to side of the road. The final tuned value was 0.15

4. Appropriate training data

Training data was chosen to keep the vehicle driving on the road. I used a combination of centre lane driving, recovering from the left and right sides of the road. Also the images were flipped and added to the training set with positive steering angle values. To further augment the training set images from left and right cameras were taken and steering angle for those images were slightly modified. As per code the vehicle would steer +0.15 on seeing left lane and -0.15 on seeing right lane. Recovery of vehicle from side lanes was the most important images taken since they prevented my vehicle to steer off the edges.

Model Architecture and Training Strategy

1. Solution Design Approach

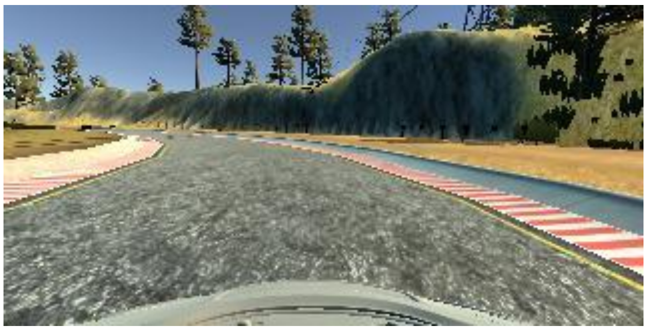
Since deep networks generally perform well to identify image, I used model used by Nvidia in their self driving car. This model 5 CNN layers and 4 fully connected layers. Image detection by this model was excellent. I only ran 3 epochs so that I don’t over fit the model.
To train the model lots of data was taken. I took more than 8000 images by running the model around the track. Then images were flipped and added to the training set. Also right and left side images were added to the training set. In the end my training set had more than 40000 images. Since it is very difficult to allocate memory to so many images together I fed the images in batch size of 100. This made training the model possible with such huge data set.
After that I splitted the images in training set and validation set to gauge the models performance. Earlier I was using mane epochs to train the model, but validation loss used to start increasing after 3 epochs, so I limited its number to 3. Loss function used for our model was mean square error, and mse value after running the model was less than 0.02 for both training and validation sets.
At the end of the process, the vehicle was able to drive autonomously around the track without leaving the road.

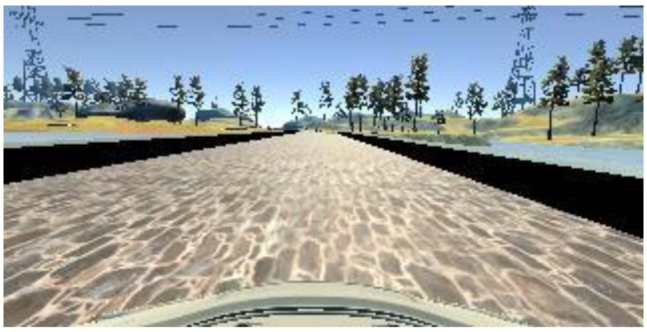
2. Final Model Architecture

The final model architecture consisted of a normalisation layer followed by cropping layer. Then there were 5 convolution neural network layers and 4 flattened layers.

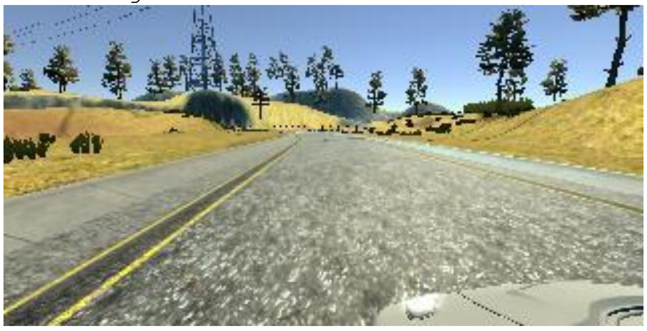
3. Creation of the Training Set & Training Process

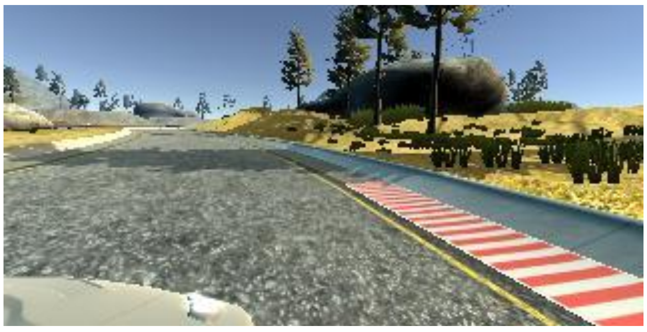
To capture good driving behavior, I first recorded two laps on track one using center lane driving. Here is an example image of center lane driving:





Right and left camera images are shown below





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 recover from side of the road.

Then I repeated this process on track two in order to get more data points. To augment the data set, I also flipped images and angles and added them to training set.

After the collection process, I had more than 40000 number of data points. I then preprocessed this data by normalizing and cropping the images.

I finally randomly shuffled the data set and put 20% of the data into a validation set.

I used this training data for training the model. The validation set helped determine if the model was over or under fitting. The ideal number of epochs was 3. I used an adam optimizer so that manually training the learning rate wasn't necessary.