Behavioral Cloning project

A. Files Submitted & Code Quality

1. Submission

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_project.pdf summarizing the results

2. 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

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.

B. Model Architecture and Training Strategy

1. Model architecture

My model is based on nvidia ones with a few modifications. It consists of a convolution neural network with 2x2 filter sizes and depths between 24 and 64.

The model includes RELU layers to introduce nonlinearity and the data is normalized in the model using a Keras lambda layer.

2. Attempts to reduce overfitting in the model

The model contains 3 dropout layers in order to reduce overfitting with a drop probability of 0.5.

The model was trained and validated on different data sets to ensure that the model was not overfitting. 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.

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.
I have also used the flipping mechanism to increase the number of images and situation.

C. Model Architecture and Training Strategy

1. Solution Design Approach

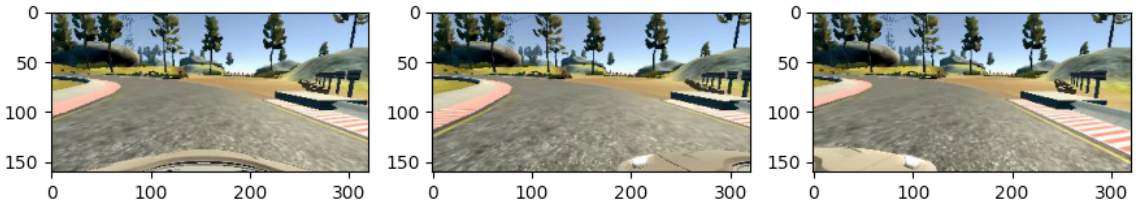
The overall strategy for deriving a model architecture was to use the different information in the videos of udacity.

My first step was to use a convolution neural network model similar to the nvidia network I thought this model might be appropriate because it was already used in autonomous driving scenario.

In order to gauge how well the model was working, I split my image and steering angle data into a training and validation set with only images from center videos and double the number of images with the flipping mechanism. 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 and add dropout mechanisms

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 in the curves. So, I have also added images from the left and right cameras to add a different vision of the road.



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 a convolution neural network with the following layers and layer sizes:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

```
=================================================================
lambda_1 (Lambda)            (None, 160, 320, 3)       0
_________________________________________________________________
cropping2d_1 (Cropping2D)    (None, 90, 320, 3)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 43, 158, 24)       1824
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 20, 77, 36)        21636
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 37, 48)         43248
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 6, 35, 64)         27712
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 4, 33, 64)         36928
_________________________________________________________________
flatten_1 (Flatten)          (None, 8448)              0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               844900
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dropout_2 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                510
_________________________________________________________________
dropout_3 (Dropout)          (None, 10)                0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 11
=================================================================
Total params: 981,819
Trainable params: 981,819
Non-trainable params: 0
```

3. Creation of the Training Set & Training Process

To capture good driving behavior, I first try to record my driving but I haven’t got the time to be enough confident with the driving and to produce enough images, so I have used the images given by udacity.
To augment the data sat, I also flipped images and angles thinking that this would increase the knowledge for right and left curves situation. I normalized the data and finally randomly shuffled the data set and put 10% 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 15. I used an adam optimizer so that manually training the learning rate wasn't necessary.