Project#3 Behavioral Cloning

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

1. Submission includes all required files and can be used to run the simulator in autonomous mode

My project includes the following files:
* DataFactory.py to do data loading work
* GenModel.py containing the script to create and train the model
* drive.py for driving the car in autonomous mode
* model_small.h5 containing a trained convolution neural network
* PROJECT3_writeup.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
```sh
python drive.py model_samll.h5
```

3. Submission code is usable and readable

The GenModel.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

1. An appropriate model architecture has been employed

My model consists of 1 convolution neural network with 5x5/3x3/1x1 filter sizes and depths between 32 and 256(model.py lines 20-29)

The model includes RELU layers to introduce nonlinearity (code line 20), and the data is normalized in the model using a Keras lambda layer (code line 63).

2. Attempts to reduce overfitting in the model

After trying of dropout, it turned out that in this model, it's better not to introduce dropout in it.

The model was trained and validated on different data sets to ensure that the model was not overfitting (DataFactory.py code line 10-16). 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 (GenModel.py line 34).

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.

Model Architecture and Training Strategy

1. Solution Design Approach

The overall strategy for deriving a model architecture was to predict driving angle. So it is a regression problem.

My first step was to use a convolution neural network model similar to the traffic sign classify model. I thought this model might be appropriate because it has several conv-layer to get transform image information to a list of abstract features. From the experience of last project, those layers could produce good features.

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 it has less feature from conv-layers, also reduce the full-connected layer number from 3 to 2.

Then I collected more data, retrain the new model.

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 into the desert. to improve the driving behavior in these cases, I collected more data at those very spots.

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 (GenModel.py lines 19-33) consisted of a convolution neural network with the following layers and layer sizes :

Convolution*32,(5x5)
MaxPooling
Convolution*64,(5x5)
MaxPooling
Convolution*128,(3x3)
MaxPooling
Convolution*256,(3x3)
MaxPooling
Convolution*32,(1x1)
Convolution*16,(1x1)
Dense*128
Dense*12
Dense*1

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 passing the corner so that make the model learn more about how to handle hard corners.

And the vehicle recovery from the side:

To augment the data sat, I also flipped images and angles thinking that this would make the model avoid bias for either direction. Also I make the left/right camera image used in training follow the suggestion by video lesson.

After the collection process, I had 53268 number of data points. I then preprocessed this data by cropping and normalization.

I 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 5 as evidenced by validation error could rise at the 6th epoch. I used an adam optimizer so that manually training the learning rate wasn't necessary.