Behavioral Cloning

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

Rubric Points

Here I will consider the rubric points (https://review.udacity.com/#!/rubrics/432/view) individually and describe how I addressed each point in my implementation.

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
- writeup_report.pdf summarizing the results
- video.mp4 show the result as a video.

2. Submission includes functional code

Using the Udacity provided simulator and drive.py (just modify the speed from 9 to 15) 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.

Model Architecture and Training Strategy

1. An appropriate model architecture has been employed

the model i use based on the network from nvidia autonomous driving showed in the tutorial which consists of 1 lamda layer for normalization, 1 cropping layer, 5 layers of convolution neural network and several fully connected layers. The whole network located between 75-104 in model.py.

2. Attempts to reduce overfitting in the model

I have tried to add dropout, batchnormalization and L2 Regularization to reduce the possibility of overfitting, but the test result in the car simulation showed a even worse result.

So, the end model contains just one dropout layers in order to reduce overfitting (model.py lines 98).

The model was trained and validated on different data sets to ensure that the model was not overfitting (in line 107,in model.fit i set validation_split=0.2 to let the code automatically split train and validation subsets). 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 (model.py line 106).

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 and flipped images.

For details about how I created the training data, see the next section.

Model Architecture and Training Strategy

1. Solution Design Approach

In order to gauge how well the model was working, I split my image and steering angle data into a training and validation set.

At first, i just use a very easy neural network and just the center images to make sure that the simulator can run successfully. The car deviates a lot and just fells off the track.

Then, i modify the neural network, enlarge the dataset and tune the parameter like batch size, epochs etc. The loss and val_loss should be watched to judge whether the underfitting/overfitting is too much or is acceptable.

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

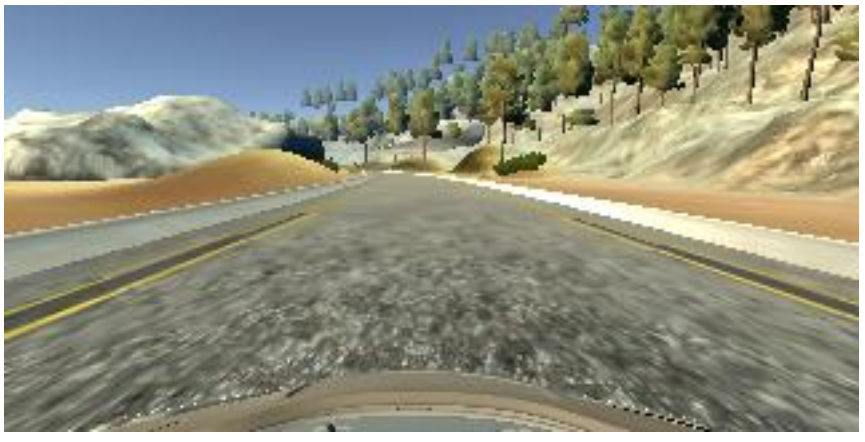
Here is a visualization of the final model architecture

Layer (type)	Output Shape	Param #
lambda_4 (Lambda)	(None, 160, 320, 3)	0
cropping2d_4 (Cropping2D)	(None, 85, 320, 3)	0
conv2d_16 (Conv2D)	(None, 81, 316, 24)	1824
activation_16 (Activation)	(None, 81, 316, 24)	0
max_pooling2d_10 (MaxPooling	(None, 40, 158, 24)	0
conv2d_17 (Conv2D)	(None, 36, 154, 36)	21636
activation_17 (Activation)	(None, 36, 154, 36)	0
max_pooling2d_11 (MaxPooling	(None, 18, 77, 36)	0
conv2d_18 (Conv2D)	(None, 14, 73, 48)	43248
activation_18 (Activation)	(None, 14, 73, 48)	0
max_pooling2d_12 (MaxPooling	(None, 7, 36, 48)	0
conv2d_19 (Conv2D)	(None, 5, 34, 64)	27712
activation_19 (Activation)	(None, 5, 34, 64)	0
conv2d_20 (Conv2D)	(None, 3, 32, 64)	36928
activation_20 (Activation)	(None, 3, 32, 64)	0
dropout_19 (Dropout)	(None, 3, 32, 64)	0
flatten_4 (Flatten)	(None, 6144)	0
dense_13 (Dense)	(None, 100)	614500
dense_14 (Dense)	(None, 50)	5050
dense_15 (Dense)	(None, 10)	510
dense_16 (Dense)	(None, 1)	11

Total params: 751,419
Trainable params: 751,419
Non-trainable params: 0

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 recovering from the left side and right sides of the road back to center so that the vehicle would learn how to recover the pose when in previous time the car tends to run out of the track.

To augment the dataset, I also use flipped images, images from left and right cameras (turning angle should be corrected by function or index)

After the collection process, I had 29820 number of data points. I then preprocessed this data by local normalization and by cropping images so as to sift out the unimportant part like sky and engine hood.

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 10 as evidenced by validation loss changes not so much. I used an adam optimizer so that manually training the learning rate wasn't necessary.

Result

the result has transformed into the file 'video.mp4', which also includes in the workspace. Please check this video.

Thank you very much for your attention!