Behavioral Cloning Project

My project includes the following files:

- model.py containing the script to create and train the model and perform the data augmentation
- drive.py for driving the car in autonomous mode
- writeup_report.pdf as a documentation
- run1.mp4 video file in which trained net complete first track

1.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
```

2. Submission code

Submission code contains functions to augment data and train model. I used several augmentation techniques such: horizontal flipping, vertical shifting using left and right cameras with steering angle correction. I also cropped images to obtain most interested region for net, avoiding unnecessary things to compute like sky or trees.

Model Architecture and Training Strategy

1. Model / Reducing overfitting / Parameters tuning

The model has been created using Keras's Sequential model. First layer is lambda layer which normalizes image. I added dropout with 0.5 and 0.25 probability after first two dense layers to prevent overfitting. I also used ELU instead ReLU. ConvNet with ELU as activation function in same cases converges faster than with ReLU.

- ConvLayer (16 x 3 x 3) / ELU / MaxPool (2x2)
- ConvLayer (32 x 3 x 3) / ELU / MaxPool (2x2)
- ConvLayer (64 x 3 x 3) / ELU / MaxPool (2x2)
- Flatten
- Dense(500) / ELU / Dropout(0.5 prob)
- Dense(100) / ELU / Dropout(0.25 prob)
- Dense(20) / ELU
- Dense(1)

I used Adam optimizer with 1e-04 learning rate and MSE as loss function.

2. Appropriate training data

To get nice generalized model we should provide good and large data set. This is sometimes very time consuming process. One way to avoid that is to first of all get small data set (in my case it was about 15000 raw images) and then use augmentation techniques to get larger data set. In Keras, training process and augmentation can be split via fit_generator function which I used in my project.

I have made 4 laps in training mode. In first two laps I have driven car very smooth around curves. Third lap was taken in "recovery mode" and in last one I focused on parts of track with different surfaces. This assures that net should recognize it during autonomous mode.

While augmenting data set first of all I picked at random image and depends on from which camera it came from I apply steering angle correction. If it was left camera I added 0.25 and if it was right camera I subtracted 0.25. Then I shift up and down image and cropped from 160x320 to 32x128. That's much less data to process. This should speed up training a little bit. I also apply some brightness changes and flipped horizontally half of images. Model should be more robust. The data is split into 80% for training and 20% kept aside as validation data.

Raw image taken from center camera

Cropped and shifted

Flipped

With changed brightness

3. Training

Due to memory restrictions I used Keras's fit_generator function to augment data on CPU and train net on GPU. I also added callback to save model every each epoch. I have used batch size of 100 for 15 epochs.

4. Testing

Testing result can be seen in run1.mp4 video. Images was taken from center camera POV. Car in autonomous mode completes first track and has driven one lap without getting out of drivable track surface. I noticed that best results (model is steadiest) appears on fastest graphic mode in simulator(25 speed target). Another thing is that car waves too much during test. I think this issue is caused by small data set or by method of data gathering. Maybe I should make same changes in net topology or add more augment techniques to make model more generalized and steady. This are 3 things which I should improve.