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 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.md or writeup*report.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.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

I try LeNet model before, but often out of track.
So I use NVIDIA model.

5 convoluation layer, 4 full connected layer and 1 dropout layer.

Data is normalized with lambda layer(model.py line 99).
Dlso use cropping layer for reduce input data and prevent irrelevance.
The only one dropout layer to prevent overfitting(model.py line 112)

2. Attempts to reduce overfitting in the model

The model contains dropout layers in order to reduce overfitting (model.py lines 112). the rate is 0.5
The model was trained and validated on different data sets to ensure that the model was not overfitting (code line 49-50). The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track.
Traning data set : 80%
Validation data set : 20%

3. Model parameter tuning

The model used an adam optimizer, so the learning rate was not tuned manually (model.py line 138).

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

1. Crop irrelevance region(sky,tree and hood of car)
2. Randomly flip half of images
3. Use multi-camera
    1. For left image, steering angle +0.2
    2. For right image, sterring angle -0.2

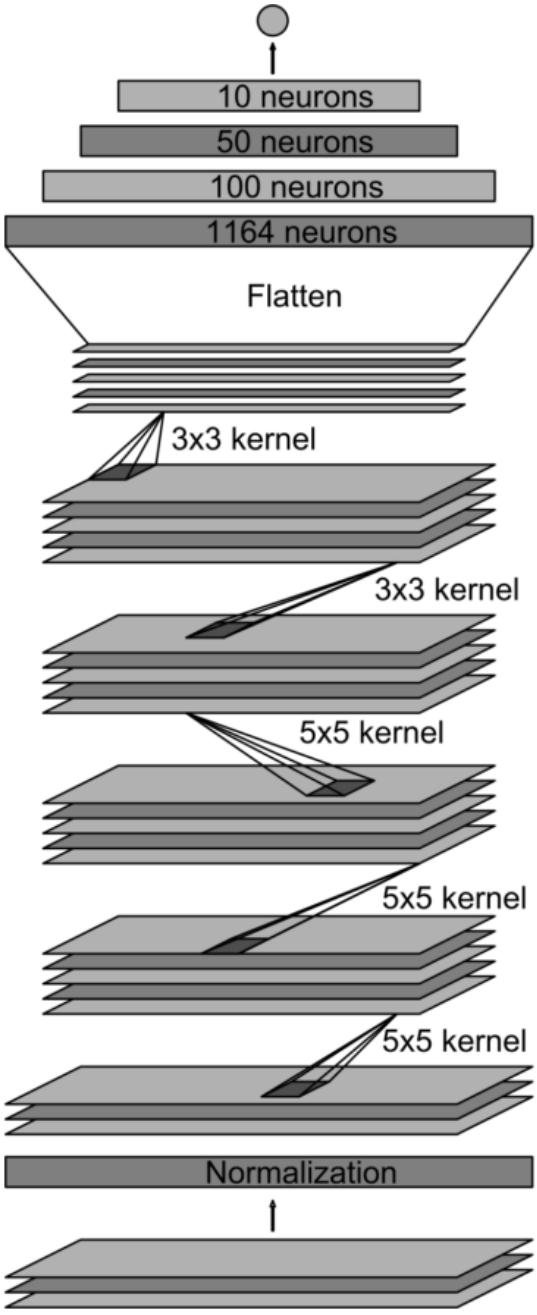
2. Final Model Architecture

The final model architecture is based on NVIDIA model (model.py lines 98-119) consisted of a convolution neural network with the following layers and layer sizes.

- Normalization for Image(input)
- Convolution 5x5, filter 24, strides 2x2, activation: elu
- Convolution 5x5, filter 36, strides 2x2, activation: elu

- Convolution 5x5, filter 48, strides 2x2, activation: elu
- Convolution 3x3, filter 64, strides 1x1, activation: elu
- Convolution 3x3, filter 64, strides 1x1, activation: elu
- Dropout 0.5
- Fully Connected, neurons 100, activation: elu
- Fully Connected, neurons 50, activation: elu
- Fully Connected, neurons 10, activation: elu
- Fully Connected, neurons 1(output)

Here is a visualization of the architecture



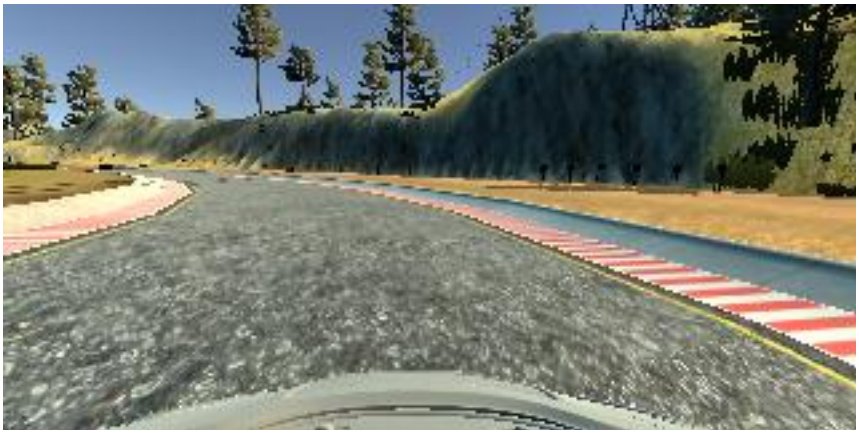
Keras Parameter

```
lambda_1 (Lambda)            (None, 160, 320, 3)       0
_________________________________________________________________
cropping2d_1 (Cropping2D)    (None, 75, 320, 3)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 36, 158, 24)       1824
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 77, 36)        21636
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 6, 37, 48)         43248
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 4, 35, 64)         27712
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 2, 33, 64)         36928
_________________________________________________________________
dropout_1 (Dropout)          (None, 2, 33, 64)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4224)              0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               422500
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_3 (Dense)              (None, 10)                510
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 11
=================================================================
Total params: 559,419
Trainable params: 559,419
Non-trainable params: 0
```

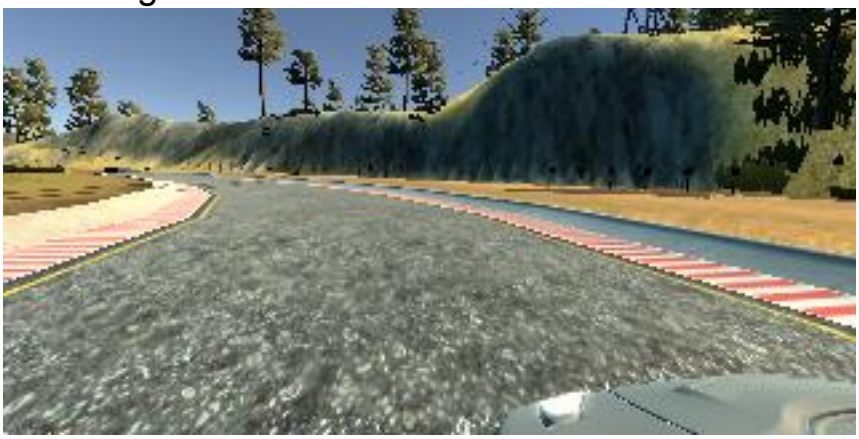
3. Creation of the Training Set & Training Process

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:

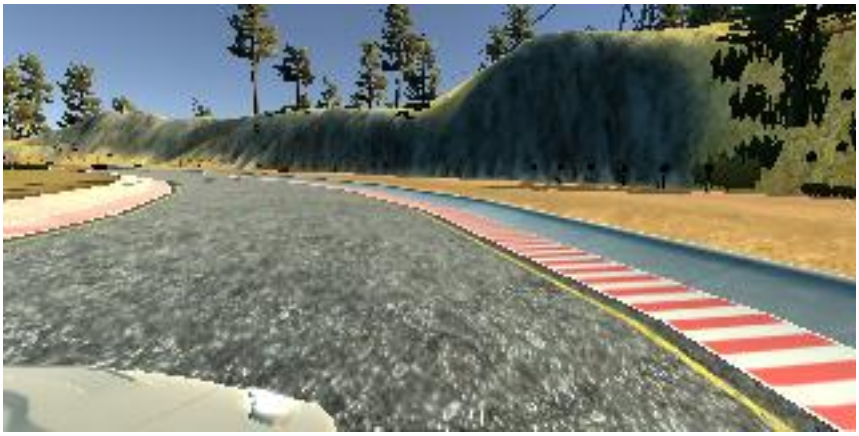
Center Image



Left Image



Right Image



For training:
I used Adam optimizer for optimizatioin by default 0.001.
ModelCheckpoint for save the model for every epoch, if validatioin loss is improved.

```
clone.py:142: UserWarning: The semantics of the Keras 2 argument `steps_per_epoch` is not the same as the Keras 1 argument `samples_per_epoch`. `steps_per_epoch` is the num
ber of batches to draw from the generator at each epoch. Basically steps_per_epoch = samples_per_epoch/batch_size. Similarly `nb_val_samples`->`validation_steps` and `val_s
amples`->`steps` arguments have changed. Update your method calls accordingly.
  verbose=1)
clone.py:142: UserWarning: Update your `fit_generator` call to the Keras 2 API: `fit_generator(<generator..., 19284, 3, validation_data=<generator..., callbacks=[<keras.ca.
.., verbose=1, validation_steps=4821)`
  verbose=1)
Epoch 1/3
2018-02-18 13:43:21.115733: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports instructions that this TensorFlow binary was not compiled to use: SSE4.1
SSE4.2 AVX
2018-02-18 13:43:21.165014: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:892] successful NUMA node read from SysFS had negative value (-1), but there must be at l
east one NUMA node, so returning NUMA node zero
2018-02-18 13:43:21.165242: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Found device 0 with properties:
name: GRID K520 major: 3 minor: 0 memoryClockRate(GHz): 0.797
pciBusID: 0000:00:03.0
totalMemory: 3.94GiB freeMemory: 3.90GiB
2018-02-18 13:43:21.165275: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1120] Creating TensorFlow device (/device:GPU:0) -> (device: 0, name: GRID K520, pci bus id:
0000:00:03.0, compute capability: 3.0)
19284/19284 [==============================] - 5216s 270ms/step - loss: 0.0078 - mean_squared_error: 0.0078 - val_loss: 0.0126 - val_mean_squared_error: 0.0126
Epoch 2/3
19284/19284 [==============================] - 5178s 268ms/step - loss: 0.0031 - mean_squared_error: 0.0031 - val_loss: 0.0121 - val_mean_squared_error: 0.0121
Epoch 3/3
19284/19284 [==============================] - 5187s 269ms/step - loss: 0.0023 - mean_squared_error: 0.0023 - val_loss: 0.0121 - val_mean_squared_error: 0.0121
```

I train 19284 data point with 3 epoch , about 4 hours.(Training on AWS GPU, run simlator on local machine)
Maybe need reduce irrelevance data.
I got the model, loss 0.0023, valdation 0.0121.
Validation is no good. seems to stuck on 0.0121.
But I try it on simulator, look good for around track 1, out of lane for track 2.