Behavioral Cloning

Writeup Template

You can use this file as a template for your writeup if you want to submit it as a markdown file, but feel free to use some other method and submit a pdf if you prefer.

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_nv_optdata_e1.h5 - containing a trained convolution neural network in nVidia Model
* writeup_report.pdf     - summarizing the results
* run_nv_optdata_e1.mp4  - video recording of vehicle driving autonomously for one lap.
```

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

```
python drive.py model_nv_optdata_e1.h5 run_nv_optdata_e1
#"run_nv_optdata_e1" - the running images will be used to generate the video.
```

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 have tried two models: 1) LeNet Model. 2) nVidia Model.

1) LeNet Model

```
def LeNet():
    print("######### In LeNet Model #########")
```

```
    model = createPreProcessingLayers()
    model.add(Conv2D(6, (5, 5),activation='relu'))
    model.add(MaxPooling2D())
    model.add(Conv2D(6, (5, 5), activation='relu'))
    model.add(MaxPooling2D())
    model.add(Flatten())
    model.add(Dense(120))
    model.add(Dense(84))
    model.add(Dropout(0.5))
    model.add(Dense(1))
    model.summary()
    return model
```

LeNet model summary is as follows:

```
    ######### In LeNet Model #########
    _________________________________________________________________
    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, 86, 316, 6)        456
    _________________________________________________________________
    max_pooling2d_1 (MaxPooling2 (None, 43, 158, 6)        0
    _________________________________________________________________
    conv2d_2 (Conv2D)            (None, 39, 154, 6)        906
    _________________________________________________________________
    max_pooling2d_2 (MaxPooling2 (None, 19, 77, 6)         0
    _________________________________________________________________
    flatten_1 (Flatten)          (None, 8778)              0
    _________________________________________________________________
    dense_1 (Dense)              (None, 120)               1053480
    _________________________________________________________________
    dense_2 (Dense)              (None, 84)                10164
    _________________________________________________________________
    dropout_1 (Dropout)          (None, 84)                0
    _________________________________________________________________
    dense_3 (Dense)              (None, 1)                 85
    =================================================================
    Total params: 1,065,091
    Trainable params: 1,065,091
    Non-trainable params: 0
```

First, I have tried the LeNet model with my driving data by simulator in "simu*data/", but it always failed to run the whole track successfully, even though I used the sample driving data with 2 epochs provided in "/opt/carnd*p3/data", it still hard to finish the whole track.

```
simu_data/simu11-clockwise, the simulated images in clockwise.
simu_data/simu12-counter-clockwise, the simulated images in counter-clockwise.
```

So I tried to found another model provided by nVidia Autonomous Car Group. now the car can finished the whole track even I have train the data for 1 epoch with sample driving data in "/opt/carnd_p3/data".

2) nVidia Model

```
def nVidiaModel():
    """    Creates nVidea Autonomous Car Group model
    """
    print("######### In nVidia Model ######### ")
    model = createPreProcessingLayers()
    model.add(Convolution2D(24,5,5, subsample=(2,2), activation='relu'))
    model.add(Convolution2D(36,5,5, subsample=(2,2), activation='relu'))
    model.add(Convolution2D(48,5,5, subsample=(2,2), activation='relu'))
    model.add(Convolution2D(64,3,3, activation='relu'))
```

```
    model.add(Convolution2D(64,3,3, activation='relu'))
    model.add(Flatten())
    model.add(Dense(100))
    model.add(Dense(50))
    model.add(Dense(10))
    model.add(Dense(1))
    model.summary()
    return model
```

nVidia model summary is as follows:

```
    ######### In nVidia Model #########
    _________________________________________________________________
    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
    _________________________________________________________________
    dense_2 (Dense)              (None, 50)                5050
    _________________________________________________________________
    dense_3 (Dense)              (None, 10)                510
    _________________________________________________________________
    dense_4 (Dense)              (None, 1)                 11
    =================================================================
    Total params: 981,819
    Trainable params: 981,819
    Non-trainable params: 0
```

2. Attempts to reduce overfitting in the model

I have tried the nVidia model for train data in /simu_data/ which contains two lap driving data: one is clockwise, and another is counter-clockwise. but it has failed to finished the whole track, I think the driving data is underfitting.

```
/simu_data/simu11-clockwise             -center/left/right images: 2143
/simu_data/simu12-counter-clockwise     -center/left/right images: 1844
Total images        : (2143+1844)x3 = 11961
Total steerings data: (2143+1844)x3 = 11961
```

So I use the sample driving data from "/opt/carnd_p3/data", which has more driving data. and I split my sample data into training and validation data. Using 80% as training and 20% as validation.

```
X_train, X_validation = train_test_split(train_data, test_size=0.2)

Total images        : 8036 x 3 = 24108
Total steerings data: 8036 x 3 = 24108
Train samples       : 24108 x 0.8 = 19286
Validation samples  : 24108 x 0.2 = 4822
```

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.

```
model.compile(loss='mse', optimizer='adam')
```

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

First, I have tried the LeNet model with my driving data by simulator in "simu*data/", but it always failed to run the whole track successfully, even though I used the Udacity sample driving data with 2 epochs provided in "/opt/carnd*p3/data", it still hard to finish the whole track.

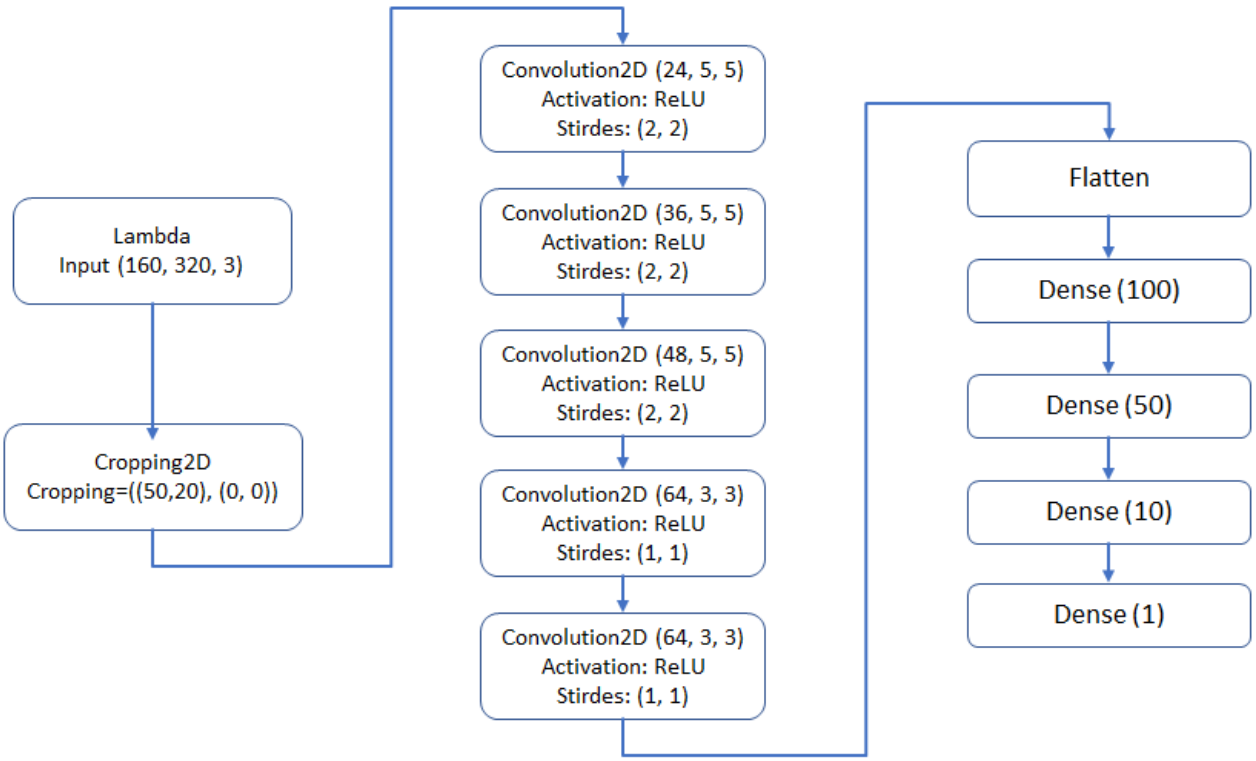
```
simu_data/simu11-clockwise, the simulated images in clockwise.
simu_data/simu12-counter-clockwise, the simulated images in counter-clockwise.
```

So I tried to found another model provided by nVidia Autonomous Car Group. now the car can finished the whole track even I have train the data for 1 epoch with sample driving data in "/opt/carnd_p3/data".

2. Final Model Architecture

The final model architecture (model.py lines 145-160) consisted of a nVidia Model with the following layers and layer sizes ...

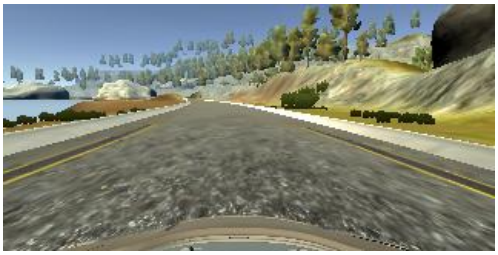
Here is a visualization of the architecture (note: visualizing the architecture is optional according to the project rubric)



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: one is clockwised, another is counter-clockwised.

Here is an example image of center lane driving:



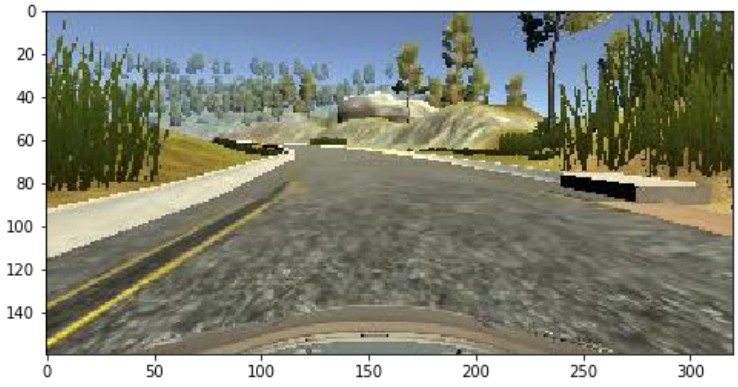
I then recorded the vehicle recovering from the left side and right sides of the road back to center so that the vehicle would learn to .... These images show what a recovery looks like starting from ... :

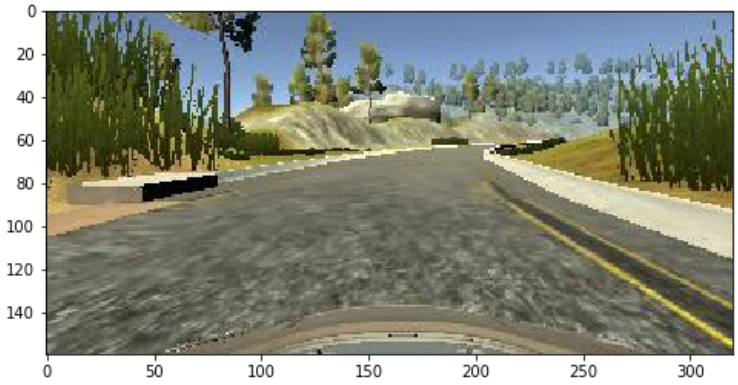
From right side back to center:


From left side back to center:


Then I repeated this process on track two in order to get more data points.

To augment the data sat, I also flipped images and angles thinking that this would ... For example, here is an image that has then been flipped:





I finally randomly shuffled the data set and put 20% of the data into a validation set.

After training the driving data, the car can be driving down the road for over one lap.