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 that 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 python3 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 consists of a convolution neural network with 5 layers of convolution followed by 3 fully connected layers. (model.py lines 18-24)

The model includes RELU layers to introduce nonlinearity, and the data is normalized in the model using a Keras lambda layer.

2. Attempts to reduce overfitting in the model

The model contains dropout layers to reduce overfitting. Dropout with keep probability of 0.8 was applied at output of 4th and 5th convolutional layer, and the first two fully connected layers.

The model was trained and validated on different data sets to ensure that the model was not overfitting (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 (model.py line 25).

4. Appropriate training data

Use of keyboard for driving the vehicle introduced a significant amount of jerking with steering angles singing drastically during turns. This led to noisy data where the vehicle would still have zero steering angle during some instances in middle of a turn. Ideal driving behaviour is much different with no jerks to the steering angle and smooth constant steering. To allow for such navigation, mouse was used.

As the lake-side track has the vehicle driving more towards the left, the data collected from regular driving is skewed towards left turns. The input dataset (driving log and images) wasaugmented artificially to reduce data skewness.

In order to generalize, data was augmented using the following methods.

- Driving the vehicle in the **reverse** direction
- **Recovery** sequences where the vehicle was made to steer to the edge of the road and brought back to the center.
- **Flipping** each image and reverting the measured steering angle

In order to train the vehicle to navigate smoothly through sharp turns, the following methods were applied

- Use of left and right camera images with **adjustment** (0.25) to the steering angle.
- **Rotating** the center camera image by a small angle (0.5 deg) whenever absolute value of steering angle execeeded 0.2 (also referred as rotation_threshold in code).

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 the approach adopeted to 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 start with a basic model, identify the pitfalls using evaluation on the example track, and augment the model and/or data accordingly.

Initial Model - Lenet

The initial approach was to use a convolution neural network model inspired from the LeNet architecture, that is known to be good at extracting appropropriate features from images (traffic signs).

In order to gauge how well the model was working, I split my image and steering angle data into a training and validation set. Although the model gave a very low validation loss (0.0071), it did not perform well when trained using the data provided as part of the project. The input dataset was then augmented with methods mentioned before. However the vehicle was not successful in making sharp turns. Note that in this experiment, only the center image was being used.

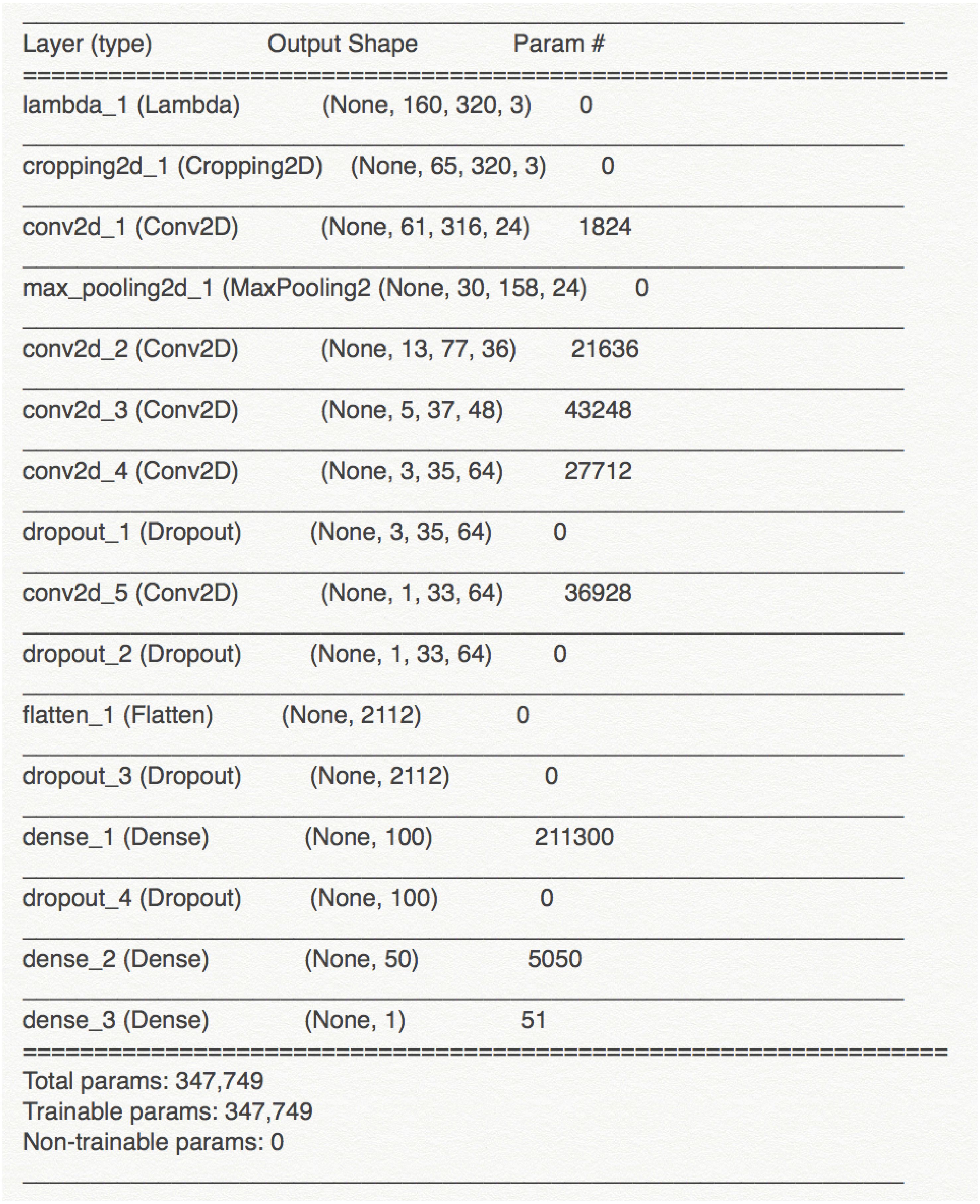
Nvidia Model

The Nvidia model is much more sophisticated with 5 convolutional layers followed by 3 fully connected layers. As it has been used before for training models for autonomous driving, it was an easy natural choice. Based on the evaluations on the track, the model was tweaked (added dropout). Please refer to the Model Architecture section for an overview of the final model that was employed after modifications.

At the end of the process, the vehicle is able to drive autonomously around the track without leaving the road.

2. Final Model Architecture

A summarized view of the model architecture is shown below.



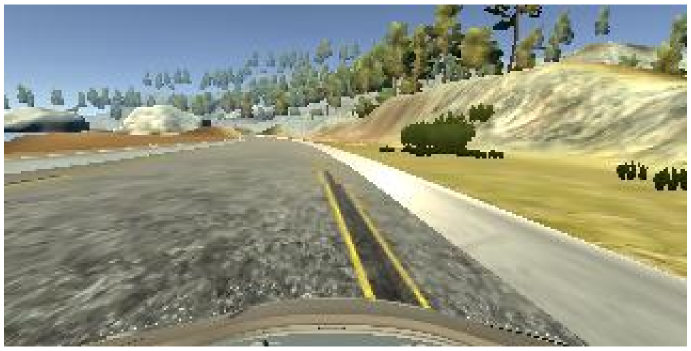
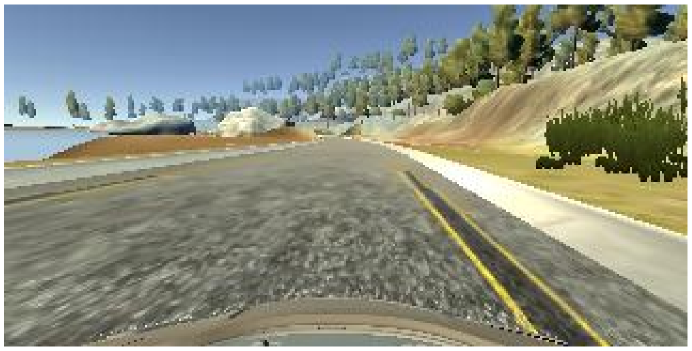
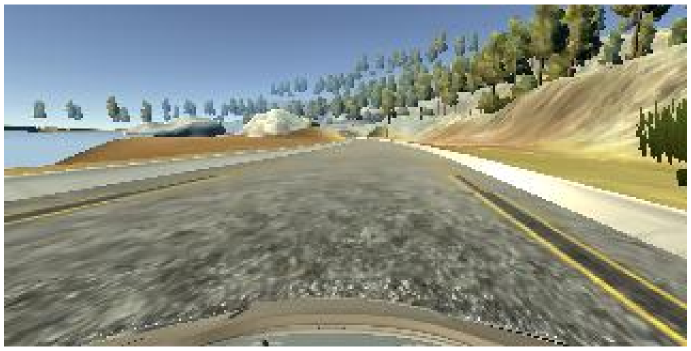
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lambda_1 (Lambda)            (None, 160, 320, 3)       0
_________________________________________________________________
cropping2d_1 (Cropping2D)    (None, 65, 320, 3)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 61, 316, 24)       1824
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 30, 158, 24)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 77, 36)        21636
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 5, 37, 48)         43248
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 3, 35, 64)         27712
_________________________________________________________________
dropout_1 (Dropout)          (None, 3, 35, 64)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 1, 33, 64)         36928
_________________________________________________________________
dropout_2 (Dropout)          (None, 1, 33, 64)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2112)              0
_________________________________________________________________
dropout_3 (Dropout)          (None, 2112)              0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               211300
_________________________________________________________________
dropout_4 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 51
=================================================================
Total params: 347,749
Trainable params: 347,749
Non-trainable params: 0
_________________________________________________________________
```

3. Creation of the Training Set & Training Process

In this section, additional details on the methods adopted to augment data are provided.

- Driving the vehicle in the **reverse** direction The vehicle was turned around and complete 5 laps on the track to create additional data that is biased towards a right turn. This woudl allow to compensate for the left-turn bias in the original track.

- **Recovery** sequences The vehicle was made to steer towards the left and right edges of the road and brought to the center again. Only the recovery sequence where the vehicle begins from an edge and reaches the center of the road was recorded.

- **Flipping** each image and reverting the measured steering angle Flipping was an obvious choice for augmenting data as the input is symmetrical. Initially flipping was enabled only for images that involved the vehicle in a turn (using flipping_threshold set as 0.3). However the threshold was later relaxed to cover all images, as it introduced instability in straighline motion of the vehicle

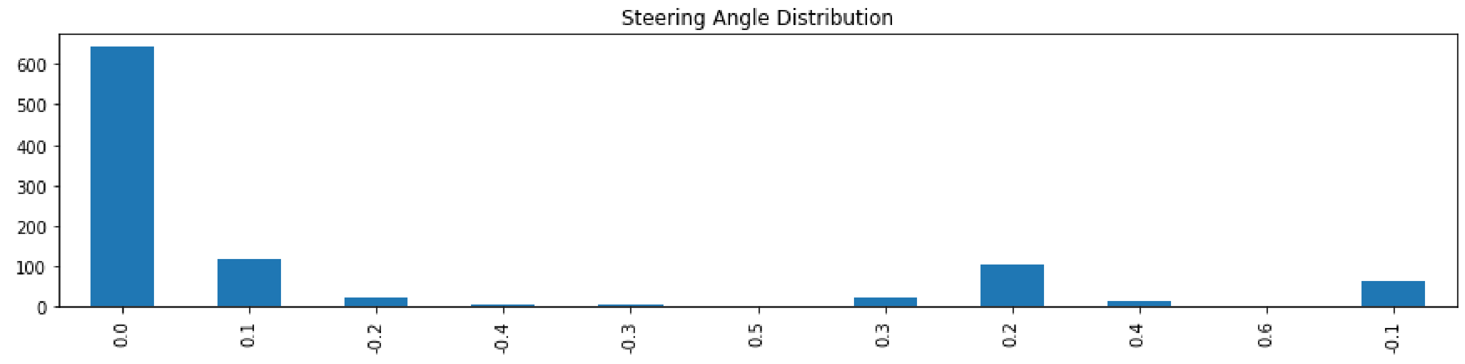
In order to train the vehicle to navigate smoothly through sharp turns, the following methods were applied

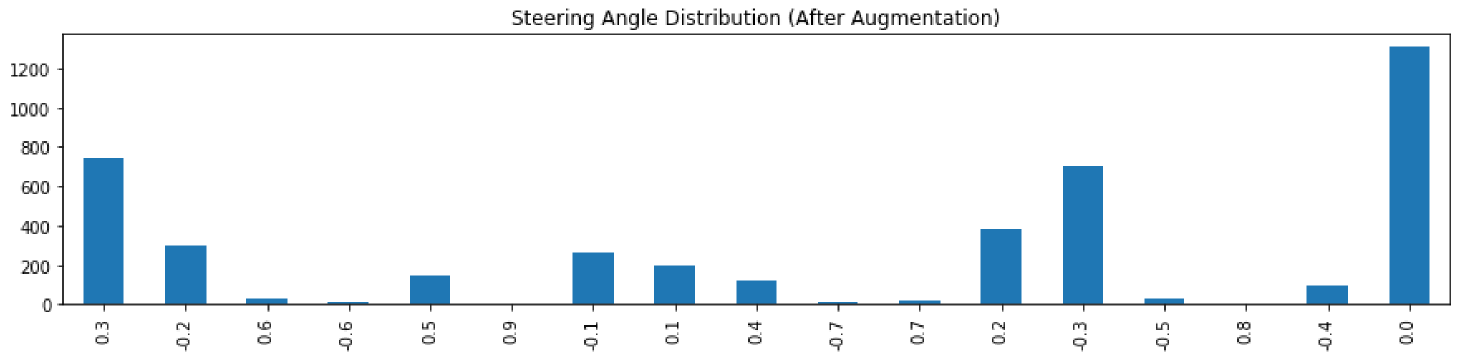
- Use of left and right camera images with **adjustment** (0.325) to the steering angle. This was the a critical adjustment as the vehicle was not unable to turn sharply and would abruptly stop turning and choosing to drive straight. The vehicle needed much powerful steering and the ability to recover if it is tending to go straight on a turn. The left and right images with adjustment to steering angle (0.325) allowed sharp turns.

- **Rotating** the center camera image by a small angle (0.5 deg) whenever absolute value of steering angle execeeded 0.2 (also referred as rotation_threshold in code). In order to further increase the samples where the car needs to turn, the samples with steering angle more than 0.5 (deg) were further rotated by a small angle (5 deg) while keeping the steering angle same as the original image.

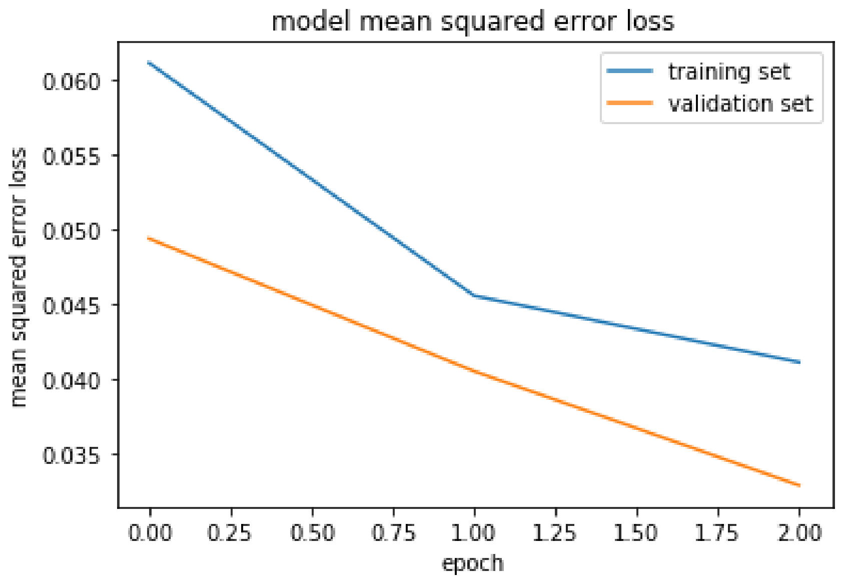
- **Translation** was applied selectively on images where the car was turning. However the translation in theory should have a similar effect to rotation on the adding more turning examples, it turned out to be not so effective. A possible reason for it would be an overfitted model with more than required turning samples. Translation was disabled in the final run.

The figures below capture the effect of artificial augmentation of samples. The images cover a randomly selected subset of 1000 samples, and shows the distribution of steering angles. Notice that initial dataset has a large skew towards a zero steering angle and only a small fraction of samples with larger steering angle.





The final augmented set data had 75k images, that was split into test and validation samples (validation_split = 0.05, or 5 % of dataset). 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 Z as evidenced by the validation loss increasing after 3 epochs. Adam optimizer so that manually training the learning rate wasn't necessary.


model mean squared error loss
training set
validation set
mean squared error loss
0.060
0.055
0.050
0.045
0.040
0.035
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
epoch