**Behavioral Cloning**

The goals of this project are:

* 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

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

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

1. **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**

My model consists of a convolution neural network with 5x5 and 8x8 filter sizes and depths between 16 and 64 (*model.py* lines 76-80)

The model includes ELU layers to introduce nonlinearity (code lines 77, 79, 83, 86), and the data is normalized in the model using a Keras lambda layer (code line 71).

1. **Attempts to reduce overfitting in the model**

The model contains dropout layers in order to reduce overfitting (*model.py* lines 82, 85).

The model was trained and validated on different data sets to ensure that the model was not overfitting (code line 101-104). The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track.

1. **Model parameter tuning**

The model used an **adam** optimizer, so the learning rate was not tuned manually (*model.py* line 89).

1. **Appropriate training data**

I have used the training data supplied by Udacity under the Resources section in Behavioral Cloning Project to train my model. The details of training are provided in the next section.

Training the model

1. Solution Design Approach

The overall strategy of deriving a model architecture was to find a model that could train well with images (preferably with 3 channels like RGB) and generalize upon them. For this project these RGB images we mainly images of roads captured by the front cameras of the vehicle in the simulator.

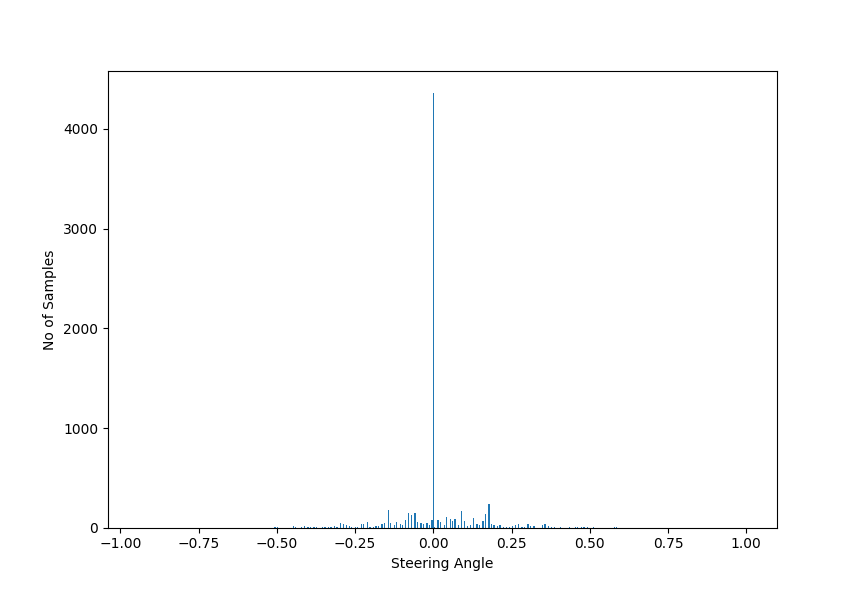
For the above purpose my first step was to use a network similar to the network used in **Comma.ai** ‘s research paper **Learning a Driving Simulator**  <https://arxiv.org/pdf/1608.01230.pdf> . The model for the neural network was adapted from **Comma.ai** ‘s GitHub repo <https://github.com/commaai/research/blob/master/train_steering_model.py> for prediction of the steering angle of the vehicle in the simulator. The reason, I think that this model is appropriate, is because:

* As mentioned in the research paper, this model worked well with ‘real world highway scenes’ and therefore it should produce good results for this simulator.
* Another reason is this steering\_angle prediction model uses ELU activation function. In previous projects I have used ReLU activation functions, but here I wanted to see how ELU works.

Given below is the network architecture used in this project

|  |
| --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Layer (type) Output Shape Param # Connected to  =========================================================================================  lambda\_1 (Lambda) (None, 160, 320, 3) 0 lambda\_input\_1[0][0]  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  cropping2d\_1 (Cropping2D) (None, 70, 320, 3) 0 lambda\_1[0][0]  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  convolution2d\_1 (Convolution2D) (None, 18, 80, 16) 3088 cropping2d\_1[0][0]  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  elu\_1 (ELU) (None, 18, 80, 16) 0 convolution2d\_1[0][0]  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  convolution2d\_2 (Convolution2D) (None, 9, 40, 32) 12832 elu\_1[0][0]  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  elu\_2 (ELU) (None, 9, 40, 32) 0 convolution2d\_2[0][0]  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  convolution2d\_3 (Convolution2D) (None, 5, 20, 64) 51264 elu\_2[0][0]  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  flatten\_1 (Flatten) (None, 6400) 0 convolution2d\_3[0][0]  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dropout\_1 (Dropout) (None, 6400) 0 flatten\_1[0][0]  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  elu\_3 (ELU) (None, 6400) 0 dropout\_1[0][0]  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dense\_1 (Dense) (None, 512) 3277312 elu\_3[0][0]  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dropout\_2 (Dropout) (None, 512) 0 dense\_1[0][0]  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  elu\_4 (ELU) (None, 512) 0 dropout\_2[0][0]  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dense\_2 (Dense) (None, 1) 513 elu\_4[0][0]  =========================================================================================  Total params: 3,345,009  Trainable params: 3,345,009  Non-trainable params: 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

The training data set distribution based on steering angles is:



We can see that most of our data is at steering angle = 0, and some between ±0.2

Initially I trained the network with this data, and using the images captured from center camera. The resultant behavior was the vehicle was poorly performing on curves and if once the vehicle moves away from the center of the road, it moves further away.

The initial parameters for training were:

|  |  |  |
| --- | --- | --- |
| EPOCHS | BATCH SIZE | LEARNING RATE |
| 5 | 128 | Adam optimizer’s default value |

To tackle these problems, I used the following strategies:

* Use the images from the left, right cameras as well. These images would help the vehicle to steer back to center of the road if it moves away from the center. A steering correction of +0.25 and -0.25 for left and right camera images were applied respectively. This technique also helped the vehicle to steer smoothly on curved roads as well.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Left camera image | Center camera image | Right camera image |

* Horizontally flipping 50% of the images and changing the sign of their steering angles during training. This would provide more examples to train the network.

|  |  |
| --- | --- |
|  |  |
| Original Image  Steering angle: **-5.05** | Horizontally flipped image  Steering angle: **5.05** |

The above two techniques were inspired from article by **Alex Staravoitau** in his article <https://navoshta.com/end-to-end-deep-learning/>

I also cropped the image from top and bottom by certain amount so that the network generalizes better on the road and unnecessary characteristics like trees & mountain doesn’t affect its decision making. This was done in **Cropping2D** layer of the network.

|  |  |
| --- | --- |
| Original Image | Cropped image |
|  | Top: **65px**  Bottom: **25px** |

Training the network with these data augmentation techniques was taking considerable time with little change in Loss and Validation accuracy. However, reducing the *batch size* to 64, helped in reducing the training time per epochs and reducing the validation loss.

This model was tested by setting the speed of the vehicle at 5mph and didn’t run out of track, however when the speed of the vehicle was increased to 9mph the vehicle went out of the road after crossing the bridge. This problem was solved by training the model for more EPOCHS.

The final parameters were:

|  |  |  |
| --- | --- | --- |
| EPOCHS | BATCH SIZE | LEARNING RATE |
| 10 | 64 | Adam optimizer’s default value |

With these values the vehicle ran smoothly across all the curves in the road.

Challenge Track

The vehicle cleared 2 or 3 curvatures on the road but when it arrived at a U-turn, it failed to make correct steering predictions and ran out of the track. Here there is scope for more fine tuning of training data. Like mentioned in the article by **Alex Staravoitau,** we can train the network with images augmented with shadows and also images vertically displaced by small offsets.