Project 3: Behaviour Cloning

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
* video.p4 showing one run in Autonomous mode around Track 1

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.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

My model consists of a convolution neural network with 2x2 filter sizes and depths between 24and 64 (model.py lines 180-186)

The model includes ELU layers to introduce nonlinearity (code line 180-192).

A Keras lambda layer is not implemented but image processing techniques along with required augmentation techniques learnt in previous projects are implemented in the form of a preprocessing pipeline. (code line 88-131).

2. Attempts to reduce overfitting in the model

The model was trained and validated on different data sets (udacity dataset, collected data set, augmented datasets) 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. The learning rate was not tuned manually but was set at the start to be 0.001

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.

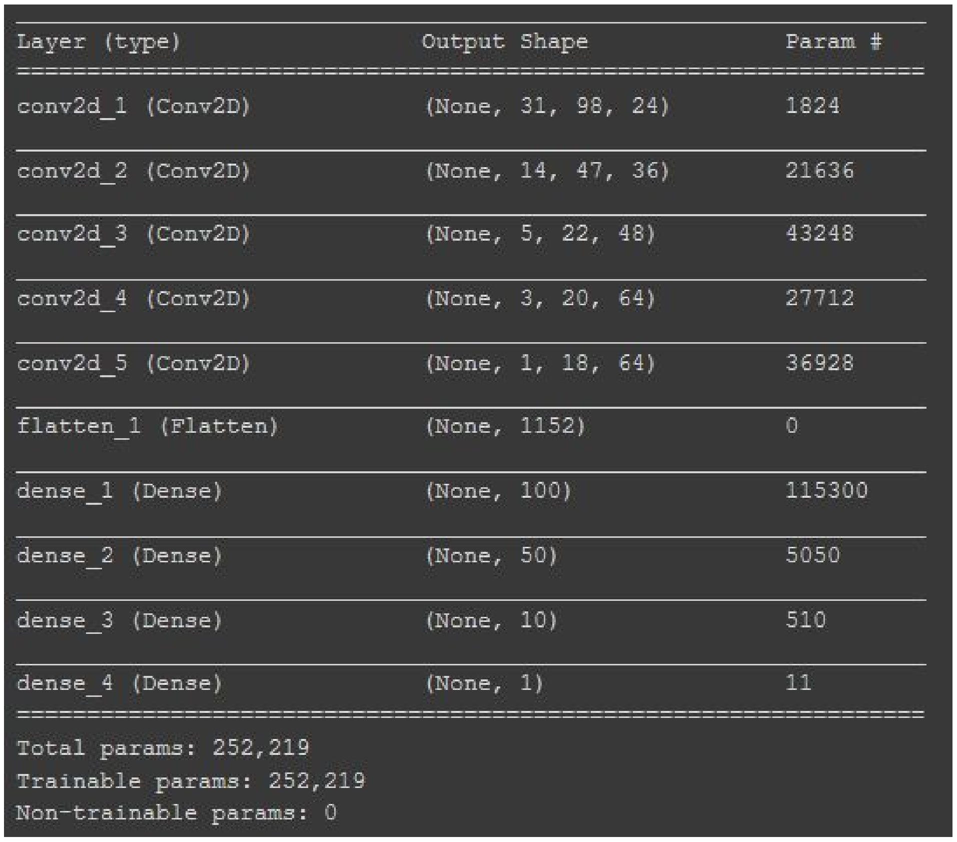
<u>**Model Architecture and Training Strategy**</u>

1. Solution Design Approach

After watching lectures, I started with a basic model to first get the preceding pipeline working and establishing connection with the simulator. After acquiring more data and developing some more image processing, I used the NVIDIA model described in the lectures to input steering values. I worked with different training samples and batch sizes as well since the time taken to complete the process was exceeding 3 hours when the NVIDIA model was implemented initially.

2. Final Model Architecture

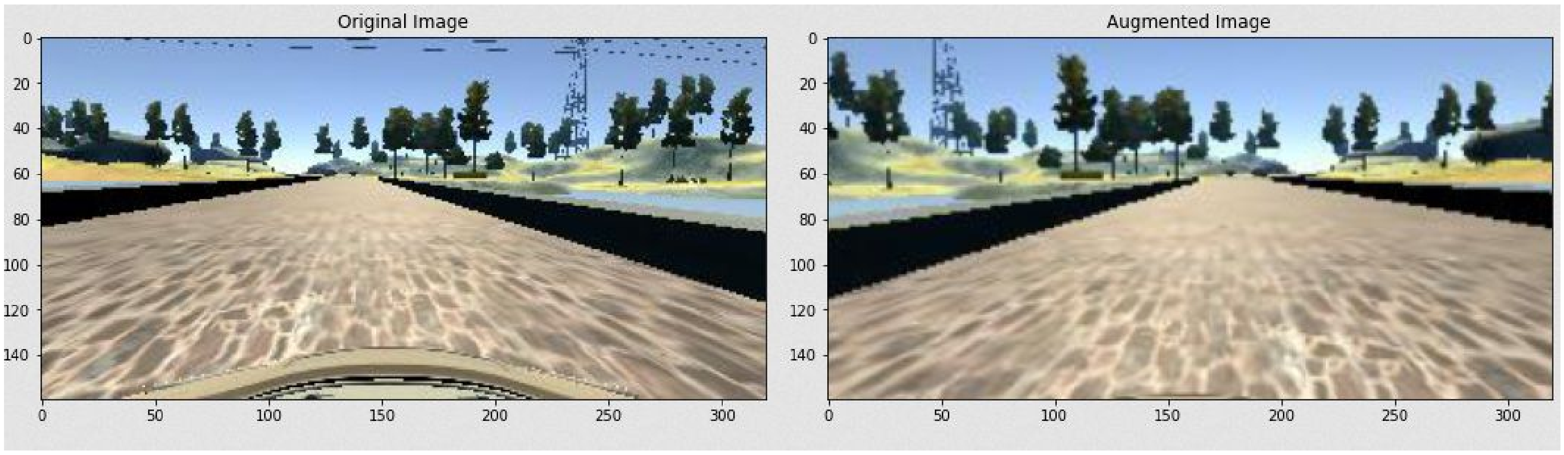
The convolutional neural net had the parameters as given in the image below-



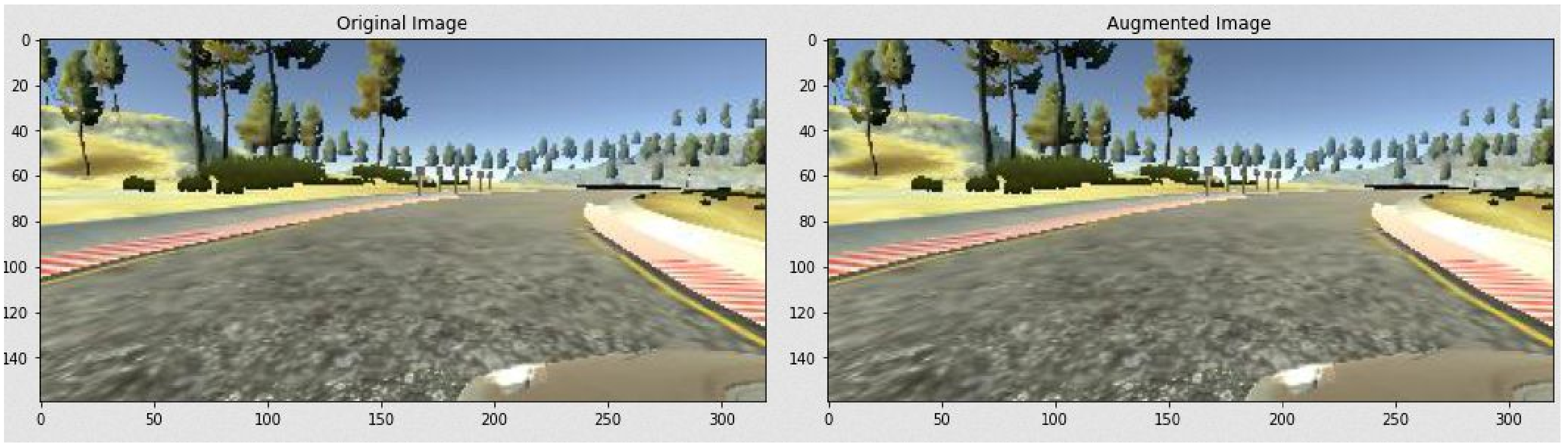
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 31, 98, 24)        1824
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 47, 36)        21636
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 5, 22, 48)         43248
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 3, 20, 64)         27712
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 1, 18, 64)         36928
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               115300
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_3 (Dense)              (None, 10)                510
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 11
=================================================================
Total params: 252,219
Trainable params: 252,219
Non-trainable params: 0
```

3. Creation of the Training Set & Training Process

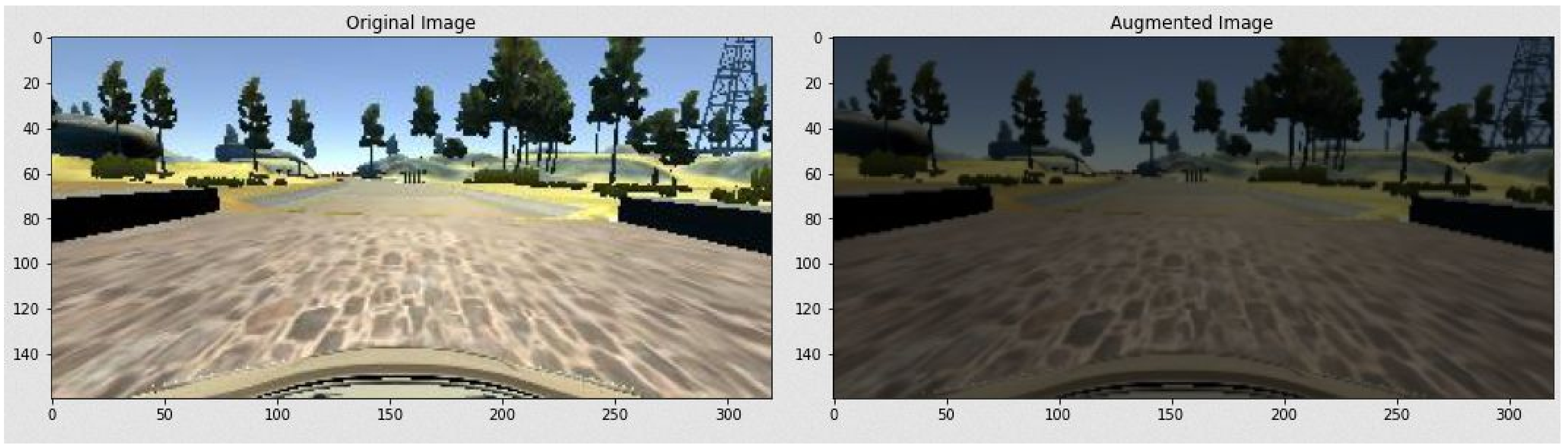
For the initial stages,I used the data provided by Udacity and also flipped the images that entered the generator. From that the vehicle showed issues like not handling turns optimally. It also showed issues on the bridge. I collected extra data which also included recovery turns to overcome these issues. In this step, I found that there was a bias towards the vehicle going straight as most of the time during training it is going straight. I randomly removed some of this data to reduce this bias (but not remove completely since most of the time it needs to go straight). During the entire development process, the data was split into training(80%) and validation(20%). Some examples are shown below-



Flipped Images

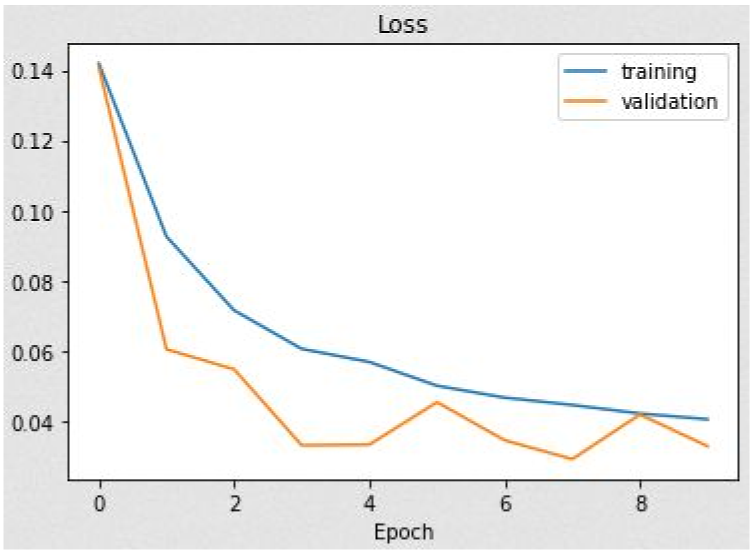


Right Steering Angle Correction



Random Brightness Adjust

For the final mode, the following loss profile was obtained for the training and validation processes. It shows both losses continuously decreasing and even though validation is a bit varying, ultimately it is not increasing. This proves that the model is not overfit.


Loss
training
validation
Epoch