**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

**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

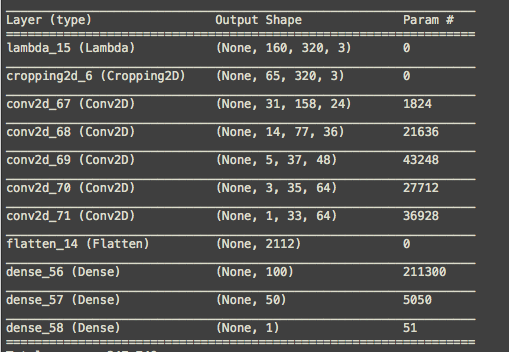
**3. Submission code is usable and readable**

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**

I have used the model shows in the figure:



My model is the convolution neural network of NVIDIA and consists in 5 convolutional layers, followed by 4 fully connected layers. The model includes RELU activators to introduce nonlinearity (code line 127-135). The model was building in Keras.

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

The model contains dropout layers to reduce overfitting (model.py lines 157, 159).

**3. Model parameter tuning**

Adam optimizer was used to set the learning rate. The loss was calculated with Mean Square Error.

**4. Appropriate training data**

I choose the left, center and right images to train correctly my car, furthermore we have applied a correction to angles.

**Model Architecture and Training Strategy**

**1. Solution Design Approach**

I have used the neural network model provided by Udacity (NVIDIA network), I started only to flip images but the car run off of the road. Then I tried to apply crop to images and this time was better but the car run off the road again.

I add resize to images (64x64) but the difference between last one and this is not so much. Finally, I removed resize and I tried to apply corrections to angles. This worked very well! Furthermore, I have used a random brightness function to increase the brightness.

To sum up, my code uses corrections angles, flip and brightness random to augment dataset.

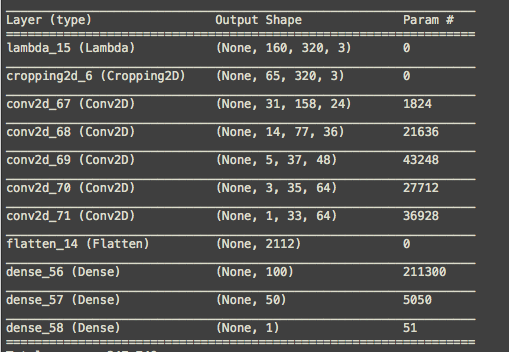
The last but not least, I built a function that take the dataset and pick up only random images to apply flip.

I run the simulator too many times and the car run off of the road but the cheat is to find the balance between the dataset size and augmentation data.

**2. Final Model Architecture**

The final model architecture (model.py code line 121-141) consists in the NVIDA network. It’s built for 5 convolutional layers and 4 fully connected layers.

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**

I used the dataset provided by Udacity, then I applied augmentation technics to augment the data set, I flipped images and angles to improve the learning. This helps to generate more data because it’s as if the car run in the opposite direction. Moreover, the model uses a random brightness function.

The dataset has also a crop technic to remove parts of the image that aren’t relevant to the train process.

