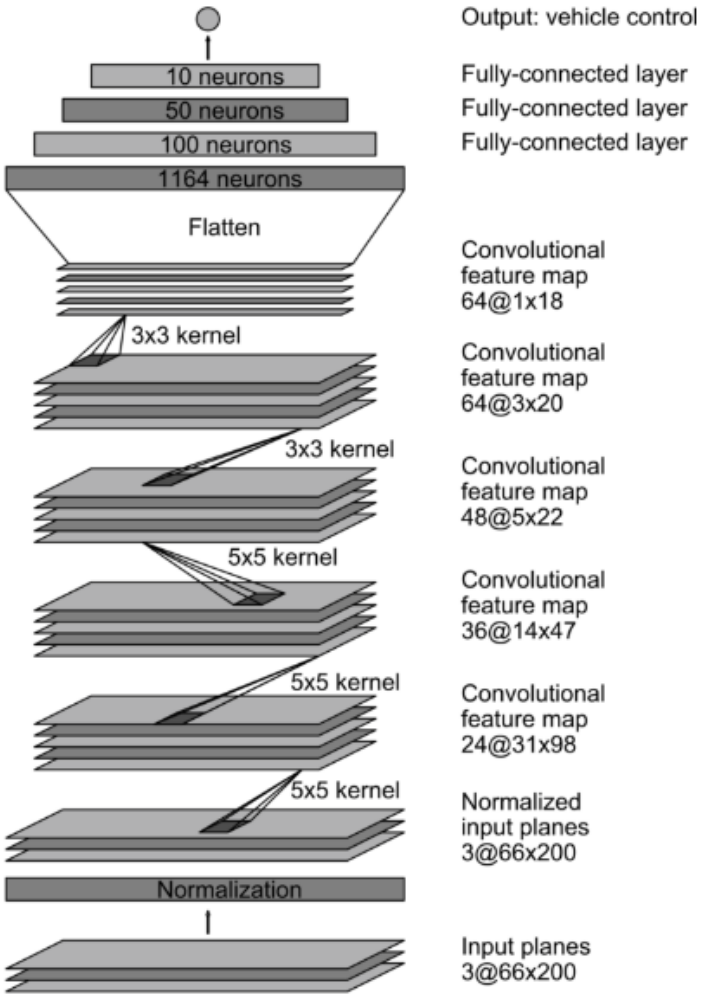
Model Architecture and Training Strategy

1. An appropriate model architecture has been employed

The model used by NVIDIA consists of 5 convolutional layers with 5x5 and 3x3 filter sizes and depths between 24 and 64 (model.py lines 68-72). That is followed by 4 fully connected layers.



The model includes RELU layers to introduce nonlinearity, and the data is normalized in the model using a Keras lambda layer (code line 66).

2. Attempts to reduce overfitting in the model

The model contains the earlyStopping callback to ensure it doesn't over fit on training data. 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 81).

4. Appropriate training data

Training data was taken from Udacity.

Strategy

1. Solution Design Approach

In order to gauge how well the model was working, I split my image and steering angle data into a training and validation set.

The final step was to run the simulator to see how well the car was driving around track one. There were a few spots where the vehicle fell off the track to improve the driving behavior in these cases, I rechecked the code and realized the cv2 module loads images ad BGR and the drive.py autonomous mode used RGB images and also changed the brightness of images randomly. At the end of the process, the vehicle is able to drive autonomously around the track without leaving the road.

2. Final Model Architecture

Layer (type)	Output Shape	Param #
cropping2d_2 (Cropping2D)	(None, 65, 320, 3)	0
lambda_2 (Lambda)	(None, 65, 320, 3)	0
conv2d_6 (Conv2D)	(None, 31, 158, 24)	1824
conv2d_7 (Conv2D)	(None, 14, 77, 36)	21636
conv2d_8 (Conv2D)	(None, 5, 37, 48)	43248
conv2d_9 (Conv2D)	(None, 3, 35, 64)	27712
conv2d_10 (Conv2D)	(None, 1, 33, 64)	36928
flatten_2 (Flatten)	(None, 2112)	0
dense_5 (Dense)	(None, 100)	211300
dense_6 (Dense)	(None, 50)	5050
dense_7 (Dense)	(None, 10)	510
dense_8 (Dense)	(None, 1)	11

Total params: 348,219
Trainable params: 348,219
Non-trainable params: 0

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

To augment the data set, I also flipped images. For example, here is an image that has been flipped and original image:


Flipped Image


Original Image

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

I used the Udacity given training data for training the model. The validation loss compared with training loss helped determine if the model was over or under fitting. The ideal number of epochs was 7 as evidenced by the earlyStopping callback.

Result:

The video run.mp4 is quite fun to watch as the car moves flawlessly on its own.