#**Behavioral Cloning**

##Writeup Template

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 filter sizes and depths between 24 and 64 (model.py lines 55-59)

The model includes RELU layers to introduce nonlinearity (code lines 55-59), and the data is normalized in the model using a Keras lambda layer (code line 53).

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

The model was trained and validated on different data sets 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, so the learning rate was not tuned manually (model.py line 66).

####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 and all the data's are augmented(flipped)

For details about how I 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 ...

My first step was to use a convolution neural network model similar to the LeNet architecture I thought this model might be appropriate because of its image recognition accuracy

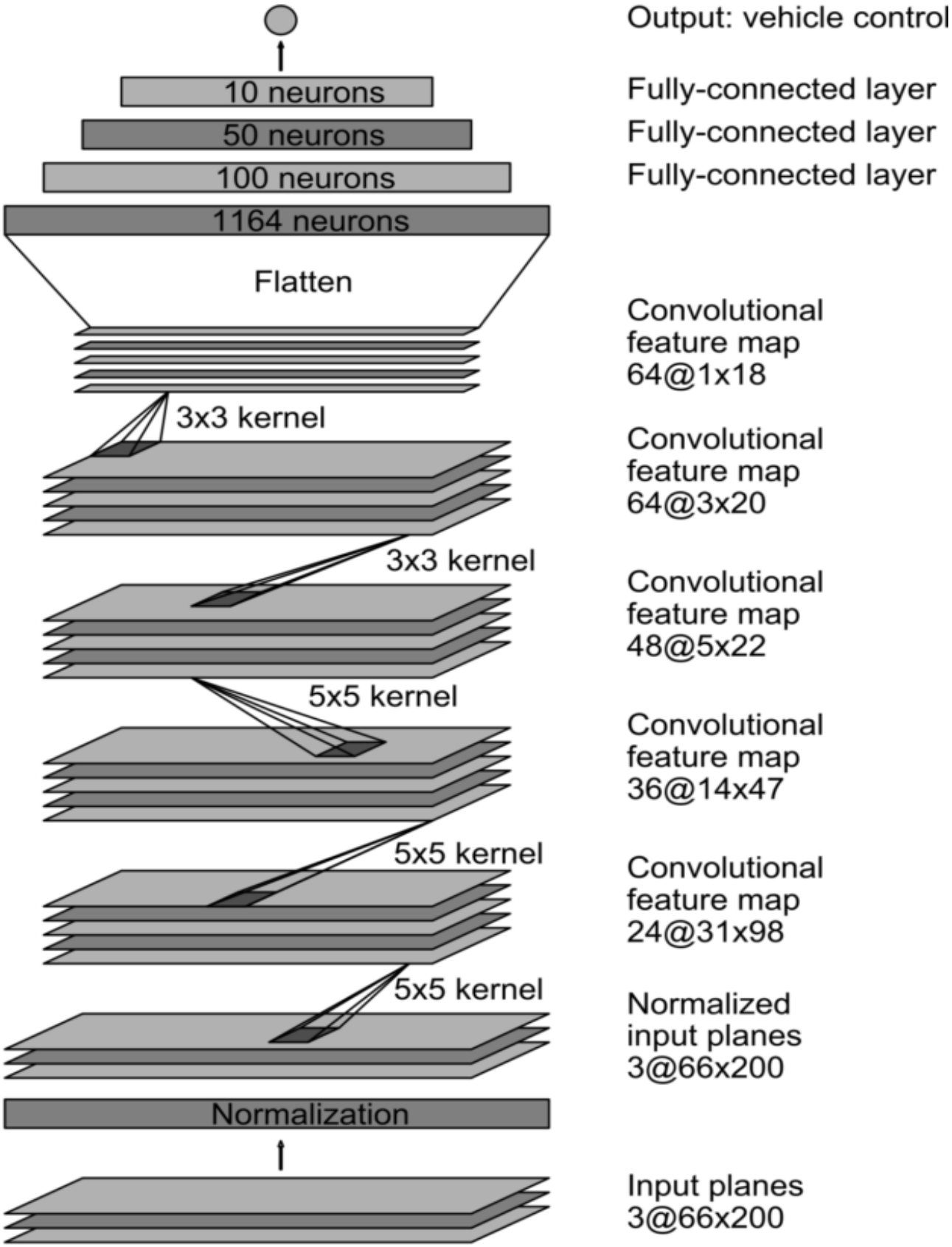
In order to gauge how well the model was working, I split my image and steering angle data into a training and validation set.

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 augmented the images as I saw the car was going in one side of the road. Then, I changed my architecture similar to Nvidia end to end architecture for self driving car

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

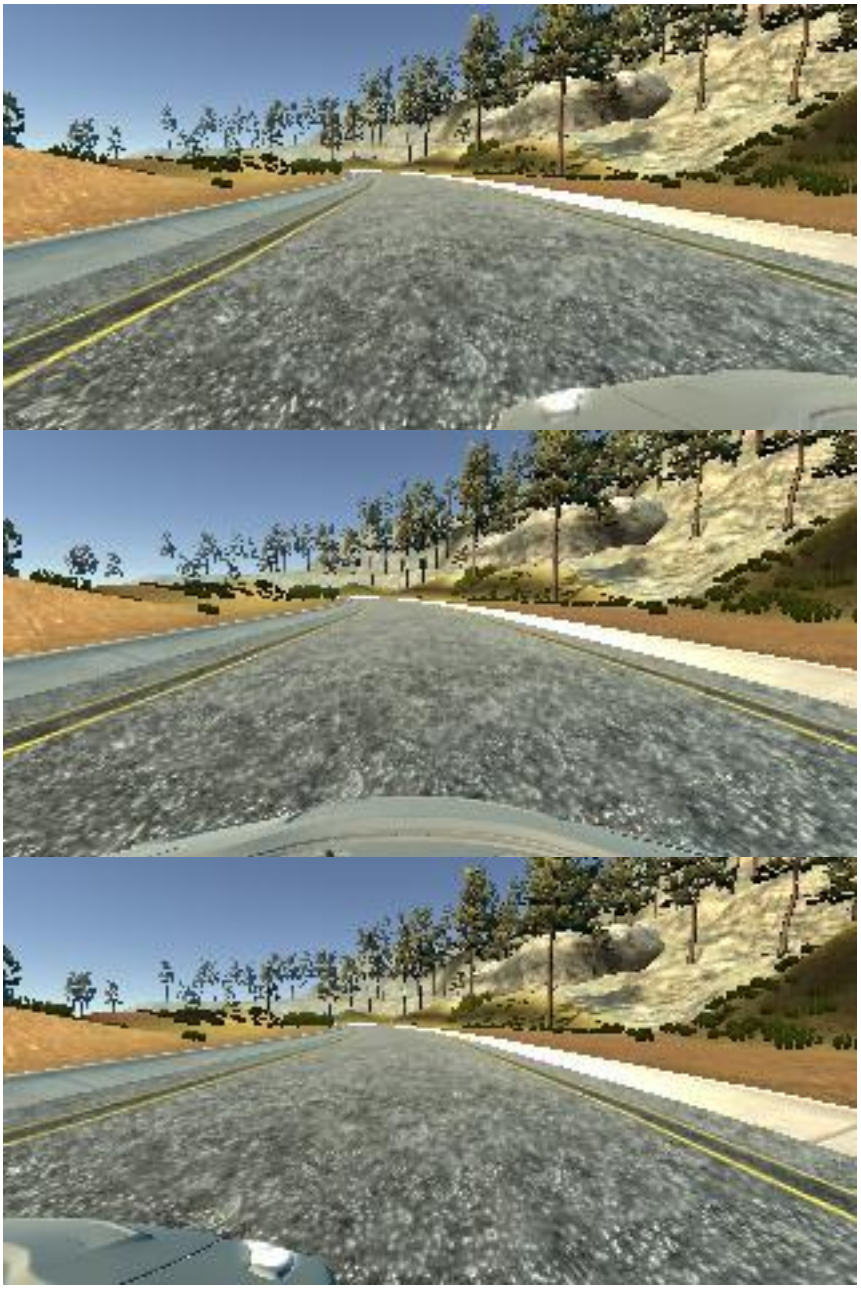
####2. Final Model Architecture

The final model architecture (model.py lines 18-24) consisted of a convolution neural network with the 5 layers and layer sizes 24, 36, 48, 64, 64 and fully connected layer with the 4 layers and layer sizes 100, 50, 10, 1. Here is a visualization of the architecture



####3. Creation of the Training Set & Training Process

To capture good driving behavior, I first recorded four laps on track one using keyboard control. Here is an example image starting from left, center and right:



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

To augment the data sat, I also flipped images and angles thinking that this would make the car to go in the center and more importantly, the data is doubled.

After the collection process, I had 42087 number of data points. I then preprocessed this data by normalization.

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

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 3. I used an adam optimizer so that manually training the learning rate wasn't necessary.