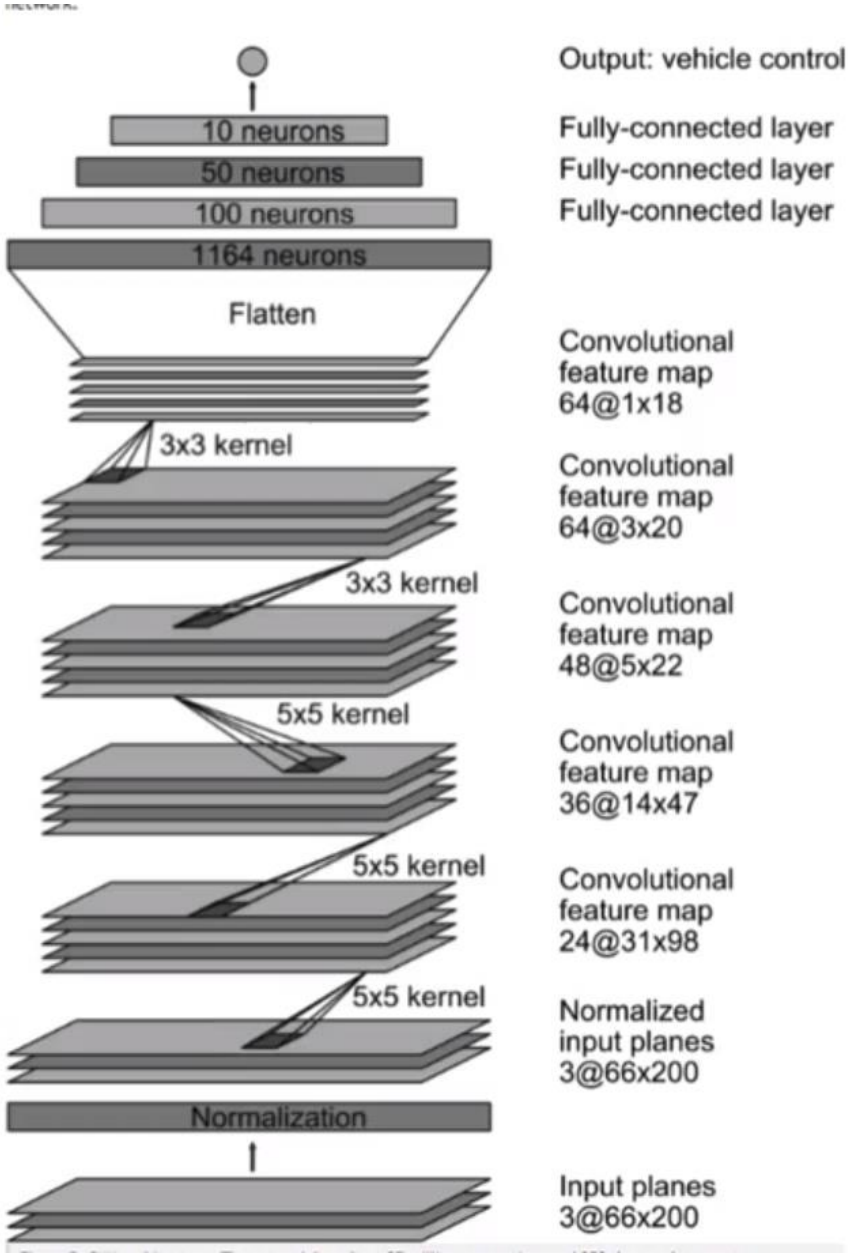
Model Architecture and Training Strategy

1. An appropriate model architecture has been employed

I use the NVIDIA model suggested by the course which looks like follow:



But It's not totally the same.

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 dropout layers in order to reduce overfitting (model.py line 70 and line 77).

The model was trained and validated on different data sets to ensure that the model was not overfitting (code line 10-16). 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 83).

4. Appropriate training data

Training data was chosen to keep the vehicle driving on the road.
I found the data provided by the course is hard to fit, so I collect data myself.
I drive counterclockwise three laps and clockwise one laps.
After training I found my model doesn't have a good performance after the bridge, so I collect that part data again.
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 taking in the camera image from the car and outputting the steering.

My first step was to use a convolution neural network model similar to the LeNt. When I found my thought works, to get better performance I choose the model that NVIDIA published. I thought this model might be appropriate because it's a real model that a company has used to train their self-driving car.

At first, I want to see how well the model can fit the data, I set the epochs 5 and the lose keep decreasing. But when I run the simulator, I found the model get a even worse performance than LeNet. I noticed that training loss and validate loss are both getting lower, I guess even the loss-data are good, but the model may has over-fitted. The model only know how to deal with the pictures it has seen before, when I run the simulator, the model don't know how to deal with the new pictures. I set the epochs to 3, and add dropout layers and It performance better.

But it seems like don't know how to turn at the first corner after bridge. I collected this part data again and train the model again.

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

2. Final Model Architecture

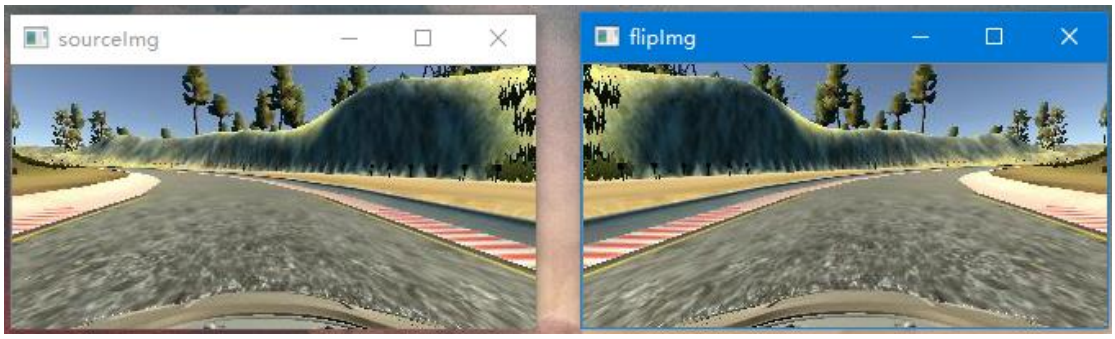
The final model architecture (model.py lines 67-81)

Layer	Shape
Normalized	lambda x: x/127.5 – 1
Cropping	cropping=((70, 25), (0, 0))
Conv2D	Filter=24, kernel_size=(5,5), strides=(2,2)
Activate	Relu
Dropout	Rate = 0.2
Conv2D	Filter=36, kernel_size=(5,5), strides=(2,2)
Activate	Relu
Conv2D	Filter=48, kernel_size=(5,5), strides=(2,2)
Activate	Relu
Conv2D	Filter=64, kernel_size=(3,3), strides=(1,1)
Activate	Relu
Conv2D	Filter=64, kernel_size=(3,3), strides=(1,1)
Activate	Relu
Flatten	

Dropout	Rate = 0.2
Fully connect	100
Fully connect	50
Fully connect	10
Fully connect	1

3. Creation of the Training Set & Training Process

At first I use the data from the course, But I found it's hard to use so I decide to collect data myself.
Taking suggestions form course, I drive counterclockwise three laps one the track one for training data.
For the generalization ability, I drive clockwise one lap.
In training, I flip all the Images and angles to augment the data set thinking the model will know better when to turn left or turn right.
As training images are loaded in BGR colors pace using cv2 while drive.py load images in RGB, I change BGR to RGB in my training code so they will have the same colors space.



And I use the left side and right side images as data that car driving back to center.
Of course I normalized the data before it's feed to the model.

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

I set the epochs 5 at first, and the loss is keeping decreasing .Everything looks good but I found the model get a bad result.
I thought the model may has over-fitting but the loss didn't tell me.
So I try to set the epoch 3 and the model perform better. But it looks like don't know how to deal with the first corner after the bridge. I collect that part data and train the data again. My model can run the car safely finally.