

Behavioral Cloning

Model Architecture and Training Strategy

1. An appropriate model architecture has been employed

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

```
110 model.add(Lambda(lambda x: x/255. - .5, # normalized data
111                   input_shape=input_shape,
112                   output_shape=input_shape))
```

Figure 1 Keras lambda layer in model.py

My model consists of a convolution neural network with 3x3, 5x5 filter sizes and depths between 24 and 64.

```
113 model.add(Cropping2D(cropping = ((65,25),(0,0)))) # take important f
114 model.add(Convolution2D(24,5,5,subsample=(2,2), activation='relu'))
115 model.add(Convolution2D(36,5,5,subsample=(2,2), activation='relu'))
116 model.add(Convolution2D(48,5,5,subsample=(2,2), activation='relu'))
117 model.add(Convolution2D(64,3,3, activation='relu'))
118 model.add(Convolution2D(64,3,3, activation='relu'))
```

Figure 2 Keras Convolution2D layer in model.py

2. Attempts to reduce overfitting in the model

The model contains dropout layers in order to reduce overfitting. 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.

```
119 model.add(Flatten())
120 model.add(Dense(100, activation='relu'))
121 model.add(Dropout(0.5)) # for preventing overfit
122 model.add(Dense(50, activation='relu'))
123 model.add(Dropout(0.5)) # for preventing overfit
124 model.add(Dense(10, activation='relu'))
125 model.add(Dropout(0.5)) # for preventing overfit
126 model.add(Dense(1))
```

Figure 3 Keras Dense & Dropout layer in model.py

3. Model parameter tuning

The model used an adam optimizer, so the learning rate was not tuned manually. In keras, adam optimizer's learningrate default value is 0.001(lr=0.001)

```
127 model.compile(loss='mse', optimizer='adam')
```

Figure 4 compile information in model.py

Model Architecture and Training Strategy

1. Solution Design Approach

The overall strategy to drive the model architecture was to reduce noise in the image and use the NVIDIA model. My first step was to use a convolution neural network model similar to the NVIDIA model. And I just added dropout and some preprocessing to the model.

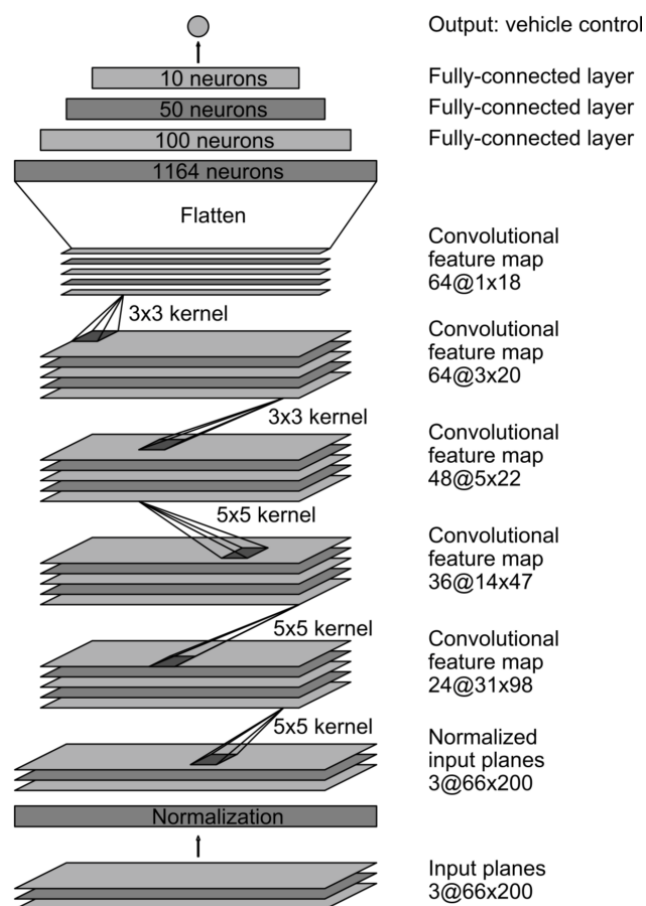


Figure 5 NVIDIA model

In order to gauge how well the model was working, I split my image and steering angle data into a training and validation set. And solving some problem while training, I used 3 camera images, converted to YUV format and added dropout to

avoid overfitting. Finally the vehicle is able to drive autonomously around the track without leaving the road.

2. Final Model Architecture

The final model architecture like below:

layer	function	param	desc
	Lamda x: $x/255. - .5$		normalization
	Cropping2D		cropping
1	Convolution2D	24, 5, 5, 2, 2	24 filter, 5x5 kernel, 2x2 strides
2	Convolution2D	36, 5, 5, 2, 2	36 filter, 5x5 kernel, 2x2 strides
3	Convolution2D	48, 5, 5, 2, 2	48 filter, 5x5 kernel, 2x2 strides
4	Convolution2D	64, 3, 3	64 filter, 3x3 kernel, 1x1 strides
5	Convolution2D	64, 3, 3	64 filter, 3x3 kernel, 1x1 strides
	Flatten		
6	Dense	100, relu	fully connected
	Dropout	0.5	don't overfit
7	Dense	50, relu	fully connected
	Dropout	0.5	don't overfit
8	Dense	10. relu	fully connected
	Dropout	0.5	don't overfit
9	Dense	1	fully connected

	compile	loss = mse, optimizer = adam	in keras, adam optimizer's learningrate default value is 0.001(lr=0.001).
--	---------	---------------------------------	--

Here is a visualization of the architecture

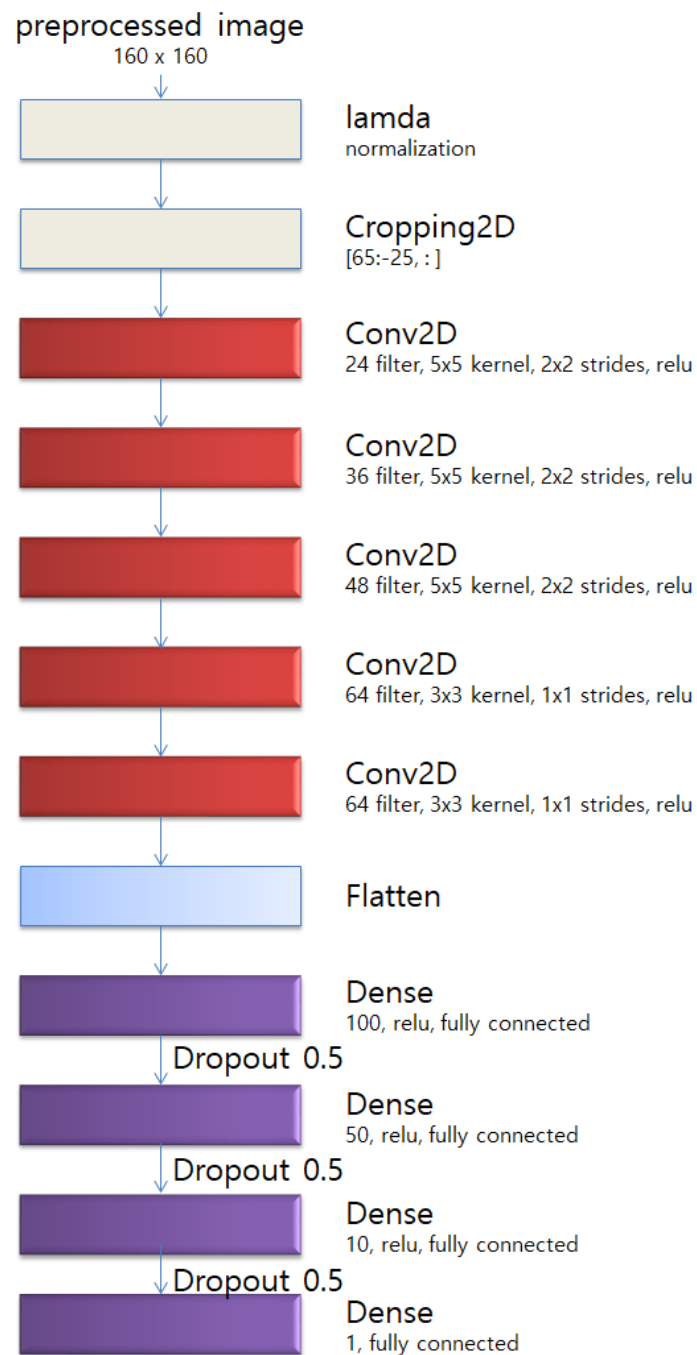


Figure 6 visualization of the architecture

3. Creation of the Training Set & Training Process

To capture good driving behavior, I recorded 65k case on track only center image.

So number of all images is 65k x 3. After recoding, we can find the data set from csv file. A, B, C columns have 3 image paths(center, left, right), and D column has center angle value.

	A	B	C	D	E	F	G
1	C:\Users\W	C:\Users\W	C:\Users\W	0	0	0	4.10E-06
2	C:\Users\W	C:\Users\W	C:\Users\W	0	0	0	5.43E-06
3	C:\Users\W	C:\Users\W	C:\Users\W	0	0	0	4.34E-06
4	C:\Users\W	C:\Users\W	C:\Users\W	0	0	0	1.78E-06
5	C:\Users\W	C:\Users\W	C:\Users\W	0	0	0	9.90E-07
6	C:\Users\W	C:\Users\W	C:\Users\W	0	0.09595	0	0.067823
7	C:\Users\W	C:\Users\W	C:\Users\W	-0.03759	0.990242	0	0.432984
8	C:\Users\W	C:\Users\W	C:\Users\W	-0.20284	1	0	1.132898
9	C:\Users\W	C:\Users\W	C:\Users\W	-0.20284	1	0	2.054177
10	C:\Users\W	C:\Users\W	C:\Users\W	-0.21256	1	0	2.725196
11	C:\Users\W	C:\Users\W	C:\Users\W	-0.21256	1	0	3.661153

Figure 7 driving_log.csv

But the drive.py code use only center images. When we use left & right images, have to add a correction value(0.2). Use the image of the path as x and each angle as y.



Figure 8 Left image, Center image, Right image

For test speed, I don't use flipped data. It exists in model.py as comments. If want, just delete string marks.

```

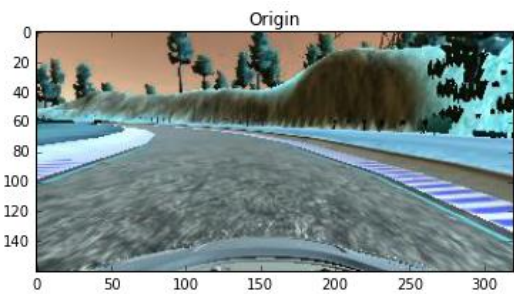
74         ...
75         # add flipped image(left & right)
76         images.append(np.fliplr(image1))
77         images.append(np.fliplr(image2))
78         images.append(np.fliplr(image3))
79
80         # create adjusted steering measurements for the side camera images
81         angles.append(-1. * center_angle)
82         angles.append(-1. *(center_angle + correction))
83         angles.append(-1. *(center_angle - correction))
84         ...

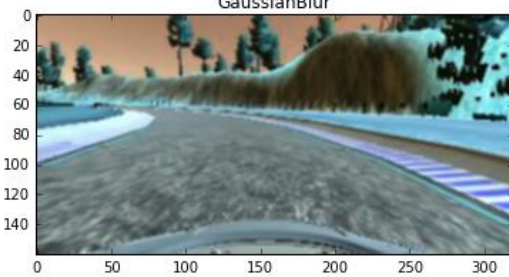
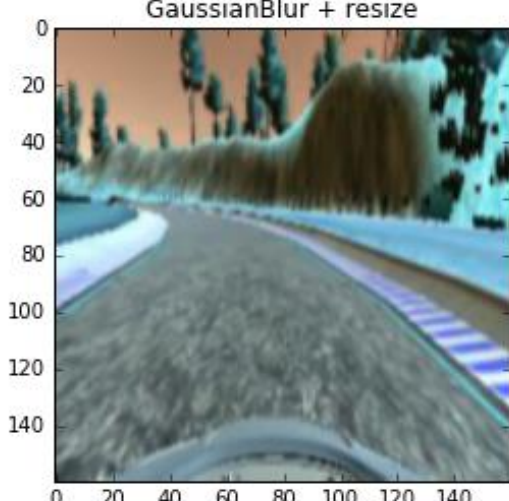
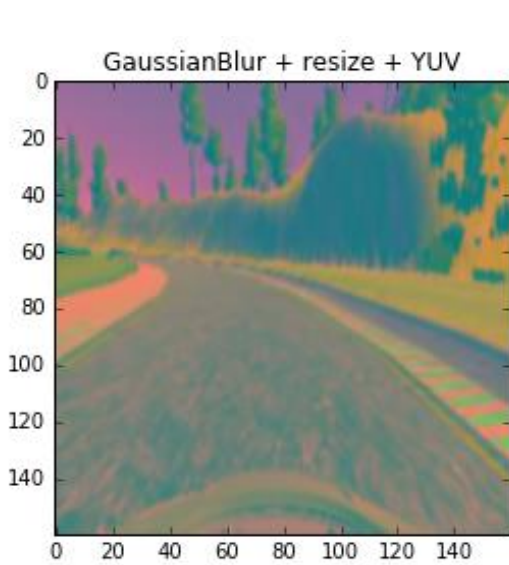

```

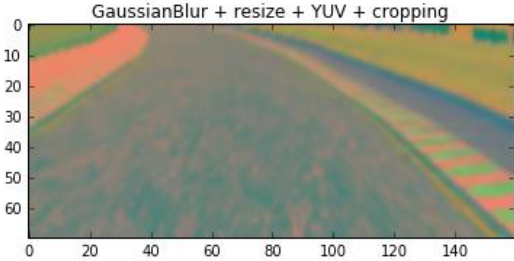
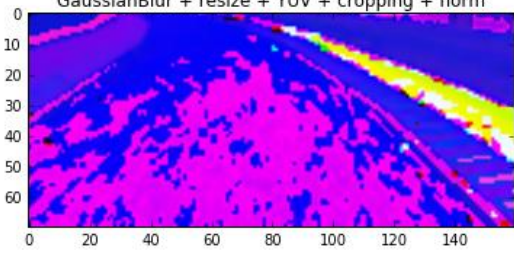
Figure 9 flip code

And I think this is most important process, Preprocess. This image has a lot of noise, we should make it to clean image. I defined the function for dataset generator, it has preprocess code. This process like this:

Processing : Gaussian Blur -> resize -> conver YUV -> cropping -> normalization

object	image	desc
Original		This is a sample image.

<p>Gaussian Blur</p>		<p>It is a widely used effect in graphics software, typically to reduce image noise and reduce detail.</p>
<p>resize</p>		<p>Because the camera look at the road from the vehicle, the road looks very short with a perspective. Using resize make the road to big and reduce the image size for performance.</p>
<p>YUV</p>		<p>Sometimes the vehicle leaves the road. I think it happened because did not properly recognize the lane as bellow. After I converted to YUV, the vehicle go thru the road very nicely.</p> 

Cropping		<p>Cropping remove the vehicle bonnet and the sky and trees. They are elements that are unnecessary for recognizing lanes. So they will be noisy.</p>
norm		<p>The YUV format data range is from 0 to 255. We can apply the normalization formula for rgb. It adds 255 and -0.5.</p>