**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.

C:\Users\kgasb\AppData\Local\Temp\enhtmlclip\Image(9).png

Figure 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.

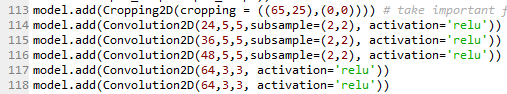


Figure Keras Convolution2D layer in model.py

**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.

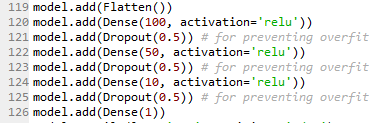


Figure Keras Dense & Dropout layer in model.py

**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)

C:\Users\kgasb\AppData\Local\Temp\enhtmlclip\Image(11).png

Figure 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.

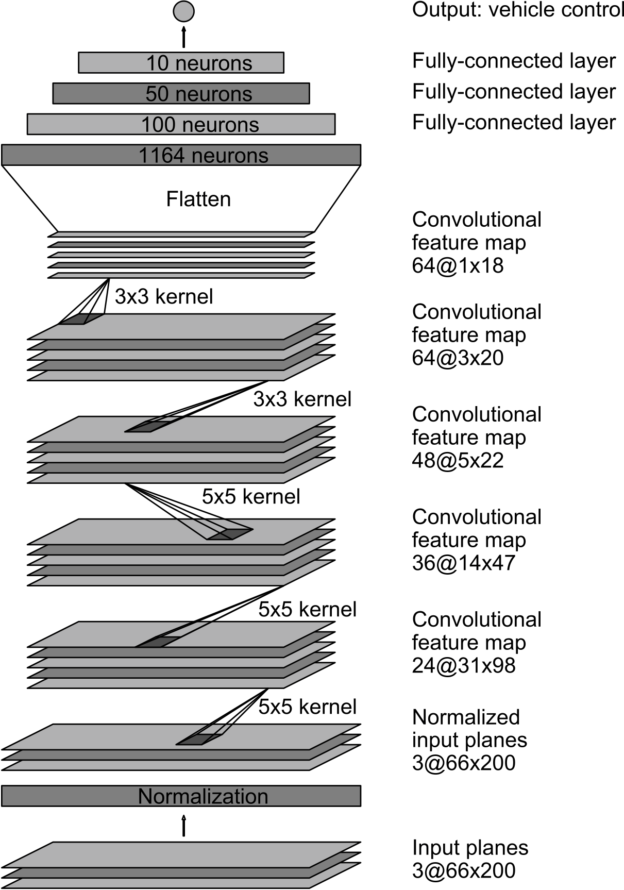


Figure NVIDIA model

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

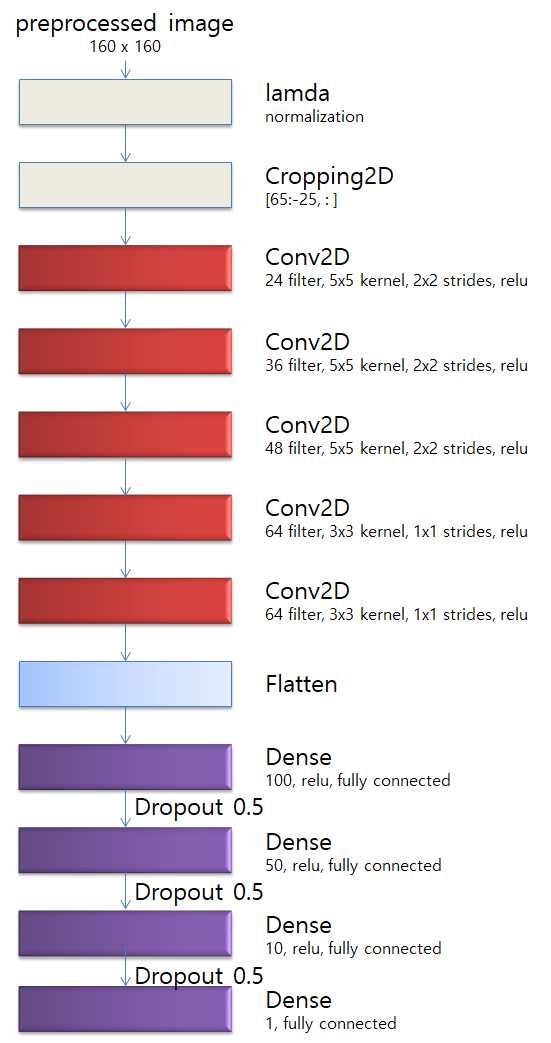


Figure visualization of the architecture

**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.

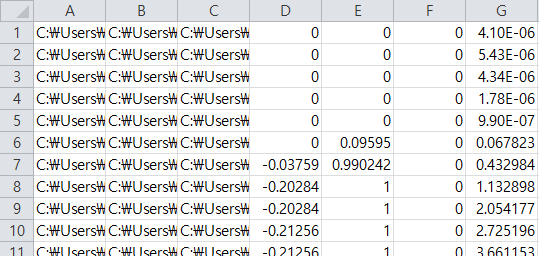


Figure driving\_log.csv

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 Left image, Center image, Rigt image

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

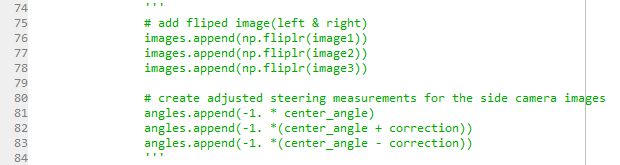


Figure flip code

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 | C:\Users\kgasb\AppData\Local\Temp\enhtmlclip\Image(3).png | This is a sample image. |
| Gaussian Blur | C:\Users\kgasb\AppData\Local\Temp\enhtmlclip\Image(4).png | It is a widely used effect in graphics software, typically to reduce image noise and reduce detail. |
| resize | C:\Users\kgasb\AppData\Local\Temp\enhtmlclip\Image(5).png | 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. |
| YUV | C:\Users\kgasb\AppData\Local\Temp\enhtmlclip\Image(6).png | 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.  C:\Users\kgasb\AppData\Local\Temp\enhtmlclip\center_2017_11_13_17_12_14_920.jpg |
| Cropping | C:\Users\kgasb\AppData\Local\Temp\enhtmlclip\Image(7).png | Cropping remove the vehicle bonnet and the sky and trees. They are elements that are unnecessary for recognizing lanes. So they will be noisy. |
| norm | C:\Users\kgasb\AppData\Local\Temp\enhtmlclip\Image(8).png | The YUV format data range is from 0 to 255. We can apply the normalization formula for rgb. It adds 255 and -0.5. |