Use Deep Learning to Clone Driving Behavior

1. **Purpose of this project:**
   A. **Use deep learning method to realize vehicle behavior cloning.**
   B. **Realize how KERAS works and generate deep learning model**
   C. **Data gathering method**
   D. **By using the trained model, make vehicle drive it alone on a track**

2. **Gathering training data set, balance data set and augmented data set**
   A. **Gathering the training data set**

      Gather data was not a simple thing since I didn't know which kind of data was good to use. For the first time, I ran around the track for 10+ laps from 3 cameras. But it didn't useful at all. So I tried to gather opposite direction data but got a same bad results. After posted my question in UDACITY forum, I realized what was wrong with my data.

   B. **Balance the training data set**

      I gather 1GB+ data using simulator. But as I mentioned before it was useless. After realized what was wrong in my data, I finished training. So, what was the core problem?

      Answer is data balance. I am not sure if it is correct to describe this phenomenon. Then how to confirm if it balanced well? By using histogram.!! It is so important , but I didn't realize before.

      If data histogram like figure 1 , then the data is totally unbalanced. Training the model with unbalanced data causes a problem: model easily overfitting to the data.

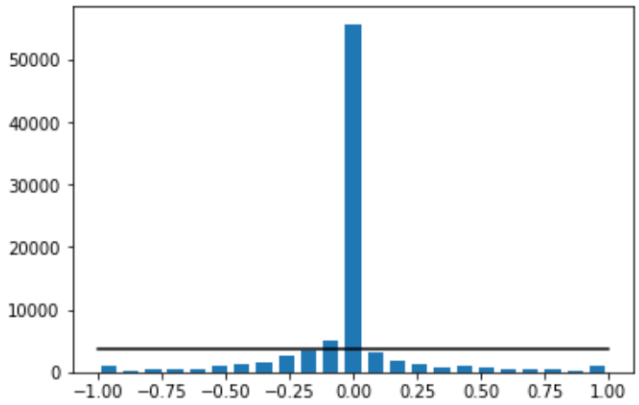

Fig. 1 histogram of images(unbalanced)

So if images histogram like figure 2, then data is balanced and wait for augmented. How to do it ? In my case, I deleted most of the steering angle which was '0'. So the ratio of '0'and other values became similarly.

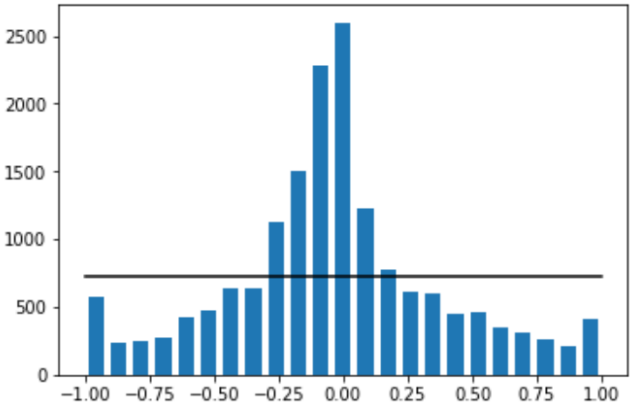

Fig. 2 histogram of images(balanced)

C. Augmented data set

Data was augmented by using following function(for generalize data & augment data). Results are showed as figure 3.

i. Randomly flip images

ii. Randomly change the brightness of the images

iii. Randomly make shadow to the images

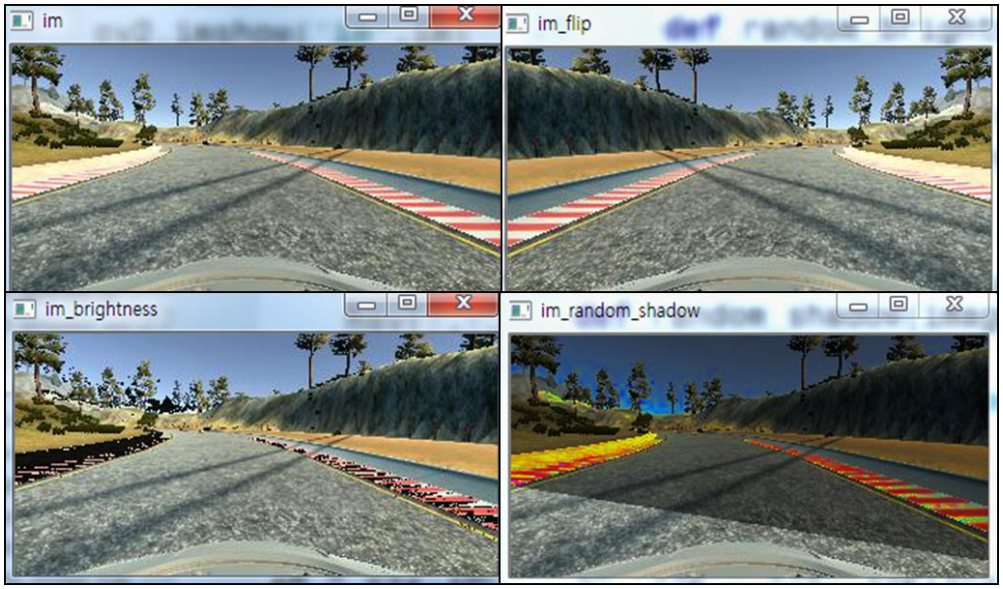

Fig. 3 data augmented images

D. Keras model structure

Model reference

Model structure is refer to the NVIDIA's paper which name is "End to End Learning for Self-Driving Cars". This is a convolutional neural network structure. In order to recognize the image correct , they use deep learning method to learning the input data(images). Like figure 4 , CNN(deep) get input from center camera and output the steering angle to vehicle control unit. By using this method, vehicle can actuate the steering angle it self.

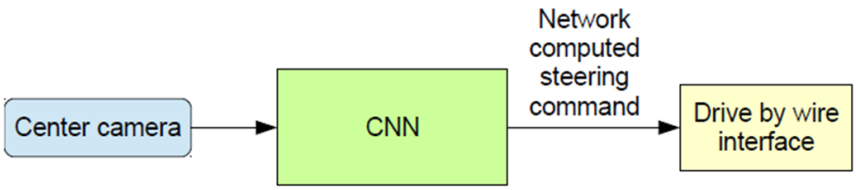

Fig. 4 CNN input and output

E. Model's layer

Figure 5 is NVIDIA's training layer. But in this project , I think it is no necessary to use such deep network. So I change the structure from

NVIDIA's structure.

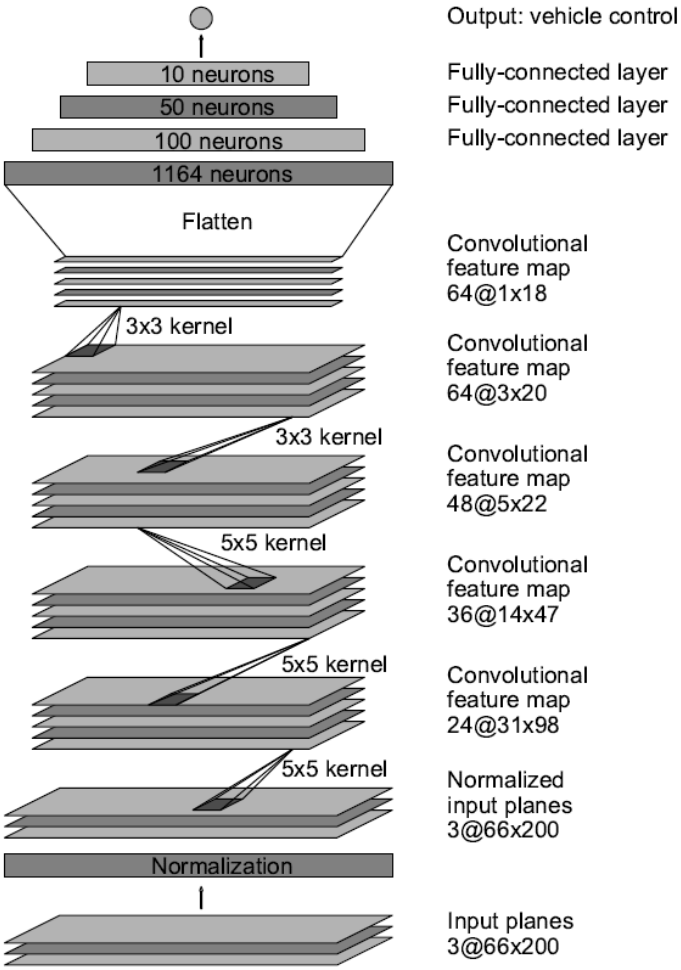

Fig. 5 NVIDIA deep learning layers

Figure 6 is my modified layers. Compare to NVIDIA's layer structure, my structure is more simple and add many dropout layers. The dropout layers' dropout probability is set as 0.2.(experimental numerical). The total parameters number is 97863 and trainable parameters is 97863. The loss function was defined as 'mse' and the optimizer is defined as ' adam' optimizer. The training epoch was 5.(I found 5 epochs is enough to training)

Layer (type)	Output Shape	Param #	Connected to
cropping2d_1 (Cropping2D)	(None, 75, 316, 3)	0	cropping2d_input_1[0][0]
lambda_1 (Lambda)	(None, 32, 100, 3)	0	cropping2d_1[0][0]
lambda_2 (Lambda)	(None, 32, 100, 3)	0	lambda_1[0][0]
convolution2d_1 (Convolution2D)	(None, 14, 48, 12)	912	lambda_2[0][0]
dropout_1 (Dropout)	(None, 14, 48, 12)	0	convolution2d_1[0][0]
convolution2d_2 (Convolution2D)	(None, 5, 22, 32)	9632	dropout_1[0][0]
dropout_2 (Dropout)	(None, 5, 22, 32)	0	convolution2d_2[0][0]
convolution2d_3 (Convolution2D)	(None, 1, 9, 48)	38448	dropout_2[0][0]
flatten_1 (Flatten)	(None, 432)	0	convolution2d_3[0][0]
dense_1 (Dense)	(None, 100)	43300	flatten_1[0][0]
dropout_3 (Dropout)	(None, 100)	0	dense_1[0][0]
dense_2 (Dense)	(None, 50)	5050	dropout_3[0][0]
dropout_4 (Dropout)	(None, 50)	0	dense_2[0][0]
dense_3 (Dense)	(None, 10)	510	dropout_4[0][0]
dense_4 (Dense)	(None, 1)	11	dense_3[0][0]

Fig. 6 my training structure

3. Training result

After define the training model, I trained data with 5 epochs. The result is show as below. Finally the model got 0.1584 loss with validation set.

```
Epoch 1/5
600/4447 [===>..........................] - ETA: 69s - loss: 0.2062
1200/4447 [=======>......................] - ETA: 30s - loss: 0.2433
1800/4447 [===========>..................] - ETA: 21s - loss: 0.2132
2400/4447 [===============>..............] - ETA: 16s - loss: 0.1934
3000/4447 [===================>..........] - ETA: 11s - loss: 0.1840
3600/4447 [=======================>......] - ETA: 6s - loss: 0.1842
4200/4447 [===========================>..] - ETA: 1s - loss: 0.1819
4800/4447 [==============================] - 50s - loss: 0.1771 - val_loss: 0.1343

Epoch 2/5
600/4447 [===>..........................] - ETA: 2s - loss: 0.2631
1200/4447 [=======>......................] - ETA: 2s - loss: 0.2063
1800/4447 [===========>..................] - ETA: 9s - loss: 0.1990
2400/4447 [===============>..............] - ETA: 9s - loss: 0.1871
3000/4447 [===================>..........] - ETA: 7s - loss: 0.1725
3600/4447 [=======================>......] - ETA: 4s - loss: 0.1726
4200/4447 [===========================>..] - ETA: 1s - loss: 0.1755
4800/4447 [==============================] - 41s - loss: 0.1722 - val_loss: 0.1449
```

```
Epoch 3/5
600/4447 [===>..........................] - ETA: 2s - loss: 0.0993
1200/4447 [=======>......................] - ETA: 2s - loss: 0.1180
1800/4447 [===========>..................] - ETA: 10s - loss: 0.1363
2400/4447 [===============>..............] - ETA: 9s - loss: 0.1414
3000/4447 [==================>...........] - ETA: 7s - loss: 0.1387
3600/4447 [======================>.......] - ETA: 4s - loss: 0.1393
4200/4447 [==========================>...] - ETA: 1s - loss: 0.1347
4800/4447 [==============================] - 42s - loss: 0.1414 - val_loss: 0.1400

Epoch 4/5
600/4447 [===>..........................] - ETA: 3s - loss: 0.1550
1200/4447 [=======>......................] - ETA: 2s - loss: 0.1621
1800/4447 [===========>..................] - ETA: 8s - loss: 0.1650
2400/4447 [===============>..............] - ETA: 9s - loss: 0.1690
3000/4447 [==================>...........] - ETA: 7s - loss: 0.1774
3600/4447 [======================>.......] - ETA: 4s - loss: 0.1743
4200/4447 [==========================>...] - ETA: 1s - loss: 0.1703
4800/4447 [==============================] - 41s - loss: 0.1680 - val_loss: 0.1550

Epoch 5/5
600/4447 [===>..........................] - ETA: 3s - loss: 0.1298
1200/4447 [=======>......................] - ETA: 2s - loss: 0.1517
1800/4447 [===========>..................] - ETA: 9s - loss: 0.1669
2400/4447 [===============>..............] - ETA: 9s - loss: 0.1575
3000/4447 [==================>...........] - ETA: 7s - loss: 0.1494
3600/4447 [======================>.......] - ETA: 4s - loss: 0.1530
4200/4447 [==========================>...] - ETA: 1s - loss: 0.1534
4800/4447 [==============================] - 42s - loss: 0.1483 - val_loss: 0.1584
```

Reference

[1]www.udacity.com

[2]Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, etc. End to End Learning for Self-Driving Cars

[3]https://github.com/windowsub0406/Behavior-Cloning

[4]https://github.com/ncondo/CarND-Behavioral-Cloning

[5]https://github.com/dyelax/CarND-Behavioral-Cloning

[6]https://github.com/naokishibuya/car-behavioral-cloning

[7]https://discussions.udacity.com/t/i-dont-know-why-my-model-cant-go-well/455549/5