Use Deep Learning to Clone Driving Behavior

1. Data Gathering and Augmentation

1.1 Data Gathering

I used the sample training data provided by udacity.

1.2 Data Augmentation

Data was augmented by using following function(for data generalization&data augmentation). Results are showed as figure 1.

- random_flip(): randomly flip images
- random_brightness(): randomly change the brightness of the images
- random_shadow(): Randomly make shadow to the images

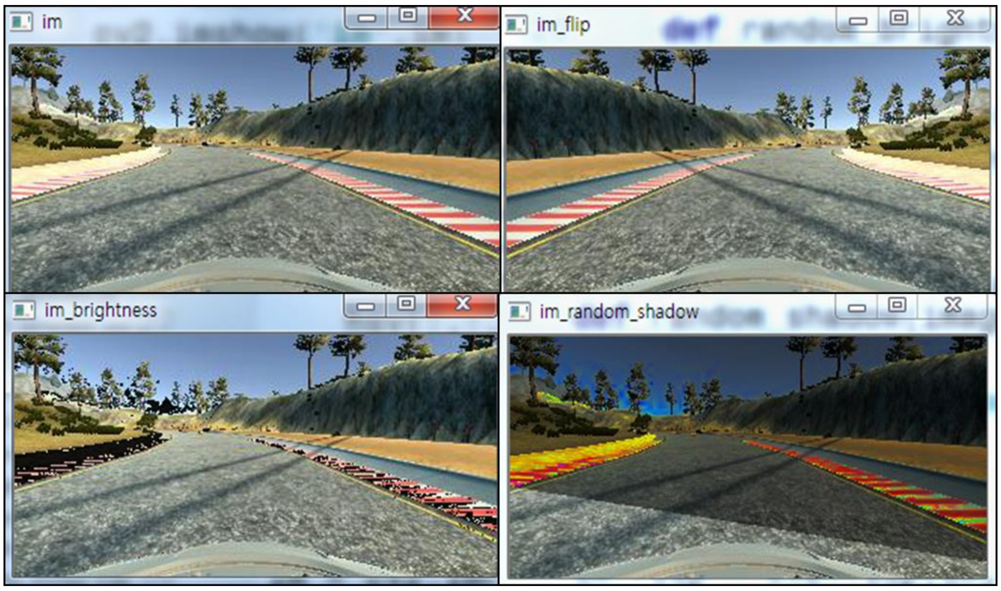


Fig. 1 data augmented images

2. Model Architecture and Training Strategy

2.1 An appropriate model architecture has been employed

The architecture of my model is refer to the NVIDIA's paper "End to End Learning for Self-Driving Cars". This is a convolutional neural network structure. In order to recognize the image correctly , they use deep learning method to learning the input data(images).As figure 4 shows, CNN(deep) get input from center camera and output the steering angle to vehicle control unit.

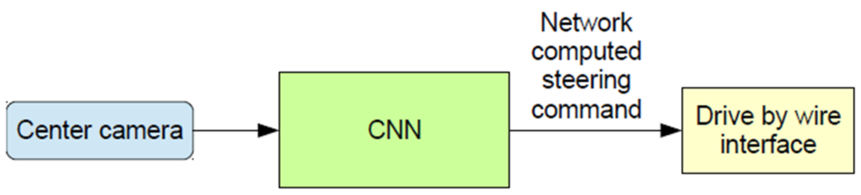


Fig. 2 CNN input and output

The architecture of model in NVIDIA's paper is shown in figure3. But in this project , I think it is unnessary to use such a deep network. So I make some changes based on its architecture.

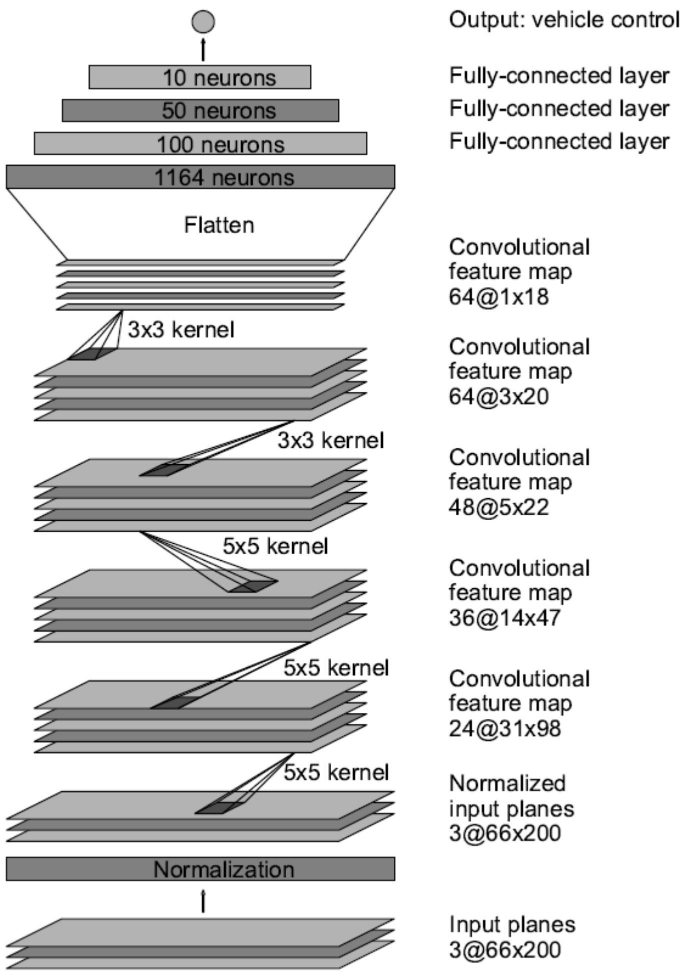
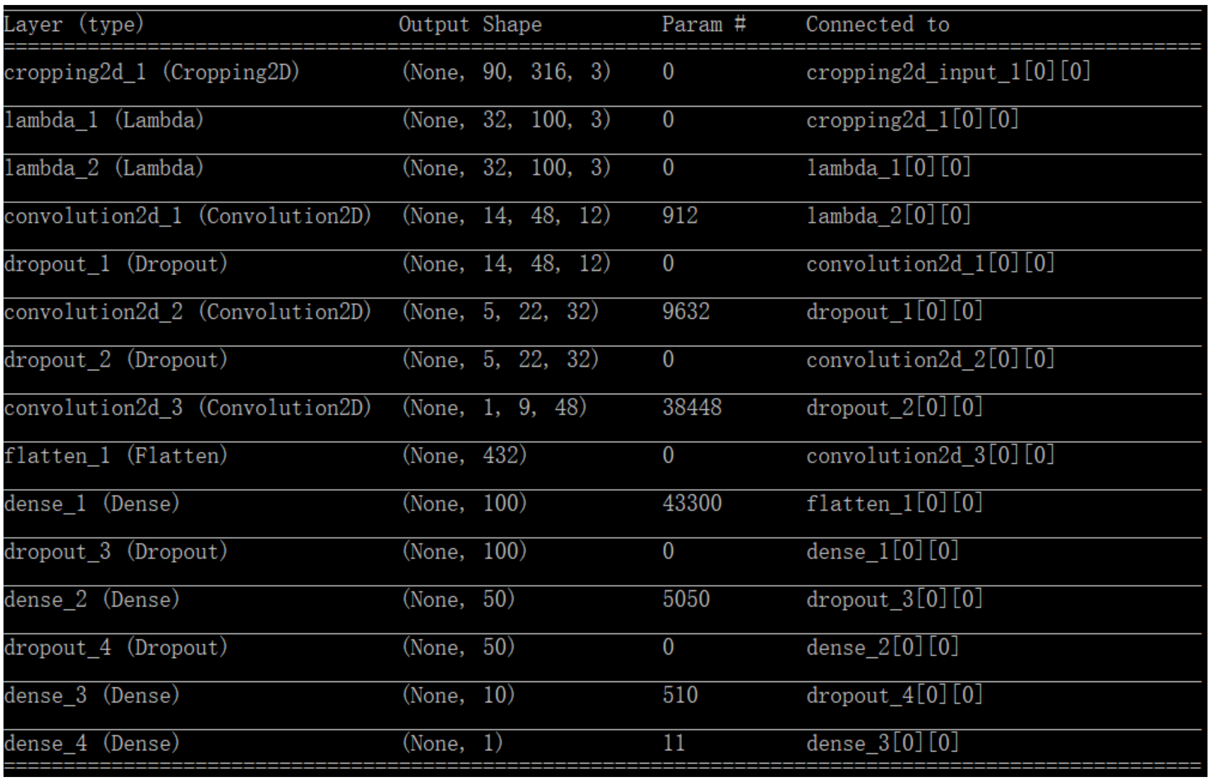


Fig. 3 model architecture in NVIDIA's paper

The architecture of my model is shown in figure4. Compared to NVIDIA's layer structure, mine has less layers but more 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, 90, 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. 4 architecture of my model

2.2 Attempts to reduce overfitting in the model

(1) The model contains many dropout layers in order to reduce overfitting (model.py lines 199,203,211,215).

(2) Data Augmentation: As shown in 1.2, I use 3 functions to augment my dataset(model.py lines 22,31,40).

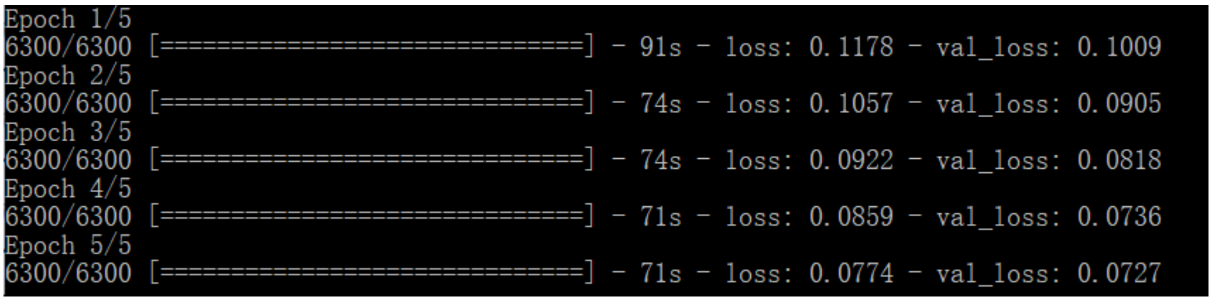
(3) Early Stopping: I only train 5 epochs and that is enough for the car to learn how to drive

2.3 Appropriate training data

In order to get training data which can keep the vehicle driving on the road, I can use a combination of center lane driving, recovering from the left and right sides of the road. But I think it will cost much time. The sample training data provided by udacity contains images gathered from left center and right cameras. From the right camera's perspective, the steering angle would be larger than the angle from the center camera. And from the left camera's perspective, the steering angle would be smaller than the angle from the center camera. At first I set the correction value to 0.2 but the car can’t keep itself in the lane in sharp bend of the road.So I reset it to 0.4 and the car can keep itself in the canter of the lane nearly in the whole track.

2.4 Training result

After defining the training model, I train data with 5 epochs. The result is show as below. Finally the model gets 0.0727 loss with validation set.



```
Epoch 1/5
6300/6300 [==============================] - 91s - loss: 0.1178 - val_loss: 0.1009
Epoch 2/5
6300/6300 [==============================] - 74s - loss: 0.1057 - val_loss: 0.0905
Epoch 3/5
6300/6300 [==============================] - 74s - loss: 0.0922 - val_loss: 0.0818
Epoch 4/5
6300/6300 [==============================] - 71s - loss: 0.0859 - val_loss: 0.0736
Epoch 5/5
6300/6300 [==============================] - 71s - loss: 0.0774 - val_loss: 0.0727
```

Fig 5. Accuracy test

We can easily find that after 5 epochs the accuracy won't obviously increase.

References:

[1] 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