Behavioral Cloning

Marcelo Eduardo Pederiva

Report

Model Architecture and Training Strategy

Model Architecture

In this project I choose to use the Nvidia Model Architecture with some modifications. I started reading an image (320,160,3) and normalized it with the Equation(1) using the Keras lambda layer.

(Pixel/177.5) – 0.5 Equation (1)

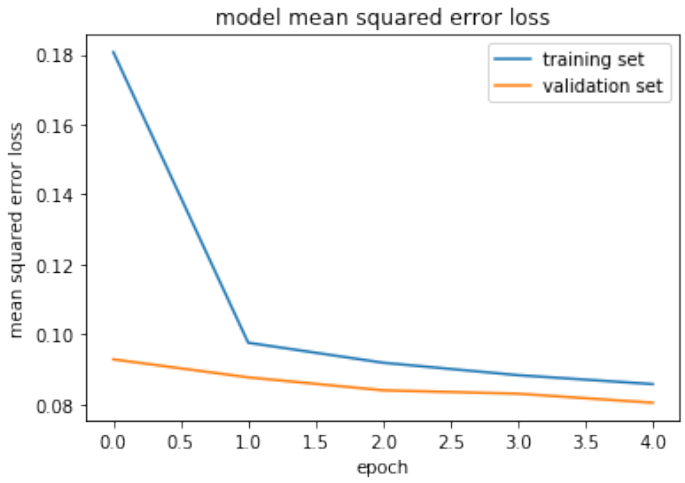
Then I cropped the image with Keras Crop layer to take out some background that could disturb the learning process, I cropped 70 pixels in upper part of the image and 25 pixel in the bottom.

With this two modifications in the input image, I started my model. As well as Nvidia model, I insert 3 convolutional layers with a 2x2 stride, 5x5 kernel and “elu” activation followed by 2 convolutional layers with a non-stride convolution, 3x3 kernel and “elu” activation. After that I put a Flatten layer and 4 Fully layers with Dropout(rate = 0.3) between then to avoid overfitting.

Layer (type)	Output Shape	Param #
lambda_1 (Lambda)	(None, 160, 320, 3)	0
cropping2d_1 (Cropping2D)	(None, 65, 320, 3)	0
conv2d_1 (Conv2D)	(None, 31, 158, 24)	1824
conv2d_2 (Conv2D)	(None, 14, 77, 36)	21636
conv2d_3 (Conv2D)	(None, 5, 37, 48)	43248
conv2d_4 (Conv2D)	(None, 3, 35, 64)	27712
conv2d_5 (Conv2D)	(None, 1, 33, 64)	36928
flatten_1 (Flatten)	(None, 2112)	0
dropout_1 (Dropout)	(None, 2112)	0
dense_1 (Dense)	(None, 100)	211300
dropout_2 (Dropout)	(None, 100)	0
dense_2 (Dense)	(None, 50)	5050
dropout_3 (Dropout)	(None, 50)	0
dense_3 (Dense)	(None, 10)	510
dense_4 (Dense)	(None, 1)	11

I used 5 epochs, “Adam” optimizer, Loss function = Mean Squared Error and Batch size = 32.

```
Epoch 1/5
301/301 [==============================] - 72s 241ms/step - loss: 0.1805 -
val_loss: 0.0929
Epoch 2/5
301/301 [==============================] - 62s 207ms/step - loss: 0.0976 -
val_loss: 0.0877
Epoch 3/5
301/301 [==============================] - 57s 190ms/step - loss: 0.0919 -
val_loss: 0.0840
Epoch 4/5
301/301 [==============================] - 56s 187ms/step - loss: 0.0883 -
val_loss: 0.0830
Epoch 5/5
301/301 [==============================] - 57s 188ms/step - loss: 0.0857 -
val_loss: 0.0805
dict_keys(['val_loss', 'loss'])
```



Training process

The training process it was a great experience because I didn’t imagine the importance and difficult to make a good training process. I tried to train only 2 laps in the center of the road and 1 lap saving the car, but I didn’t got results.

I got success using the Udacity data set, but I still wanted to make a good training data by myself.

After a lot of types of training I result to make the follow approach:

- 5 laps with the vehicle in the center of the road
- 1 lap near from the right lane (between the 2 lanes)
- 1 lap near from the left lane (between the 2 lanes)
- 1 lap saving from the right lane
- 1 lap saving from the left lane

Figure 1 – Vehicle in the center of the road

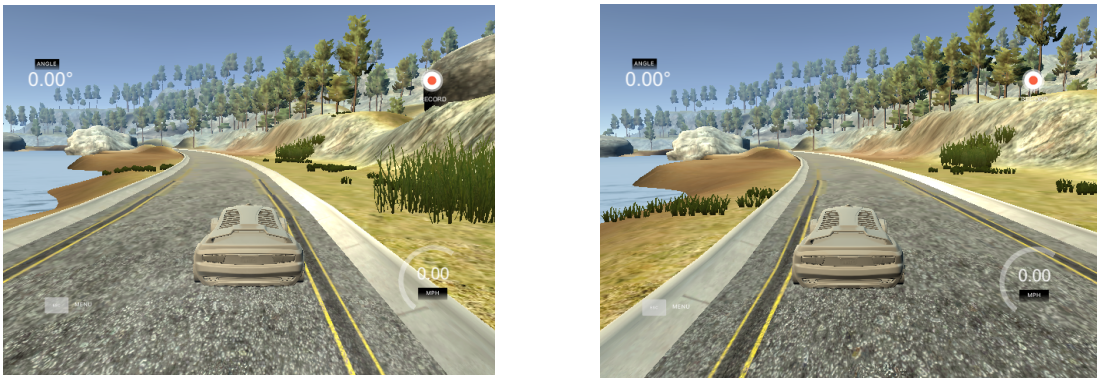


Figure 2 – Vehicle near from the right lane(left image), Vehicle near from the left lane(right image)

Figure 3 – Saving from the right lane (left image), saving from the left lane (right image)

Image Data

I used the center, left and right image from the training mode to increase the accuracy of my model. I made an adjustment in the angles of the left and right images by (angle of the center image +0.4) in the left and -0.4 in the right. Furthermore, I flipped the image and the angle to had twice the image data that I had before.

Result

Finally my model results in a run of the car in the center of the road practically all the time. In the run, the car save from passing the right lane once, but the rest of the lap the car stayed stable between the two lanes of the road.

In this project I tried to execute the Nvidia process, failing in the size and YUV color parameters of the input image. Maybe this variables would got a better performance of the car at the end, but in other hand, my model got great result.