Model

July 1, 2020

In [ ]:
```
# adding left and right camera images, and adding cropping the image from top and botton
# nvidia end-to-end learning
import csv
import cv2
import numpy as np
import glob
import keras
from keras.models import Sequential
from keras.layers import Flatten, Dense, Lambda
from keras.layers.convolutional import Convolution2D, Cropping2D
from keras.layers.pooling import MaxPooling2D

lines = []
with open('./mydata/driving_log.csv') as csvfile:
    reader =   csv.reader(csvfile)
    for line in reader:
        lines.append(line)
images = []
measurements = []

#print(lines[0])
#exit()

#imagess = glob.glob('./mydata/IMG/center*.jpg')
#for idx , fname in enumerate(imagess):
    #image = cv2.imread(fname)
    #images.append(image)


for line in lines:
    for i in range(3):
        source_path = line[i]
        tokens = source_path.split('/')
        #print(tokens)
        #exit()
        filename = tokens[5]
        #print(filename)
        #exit()
```

```
        local_path = "./mydata/IMG/" + filename
        #print(local_path)
        #exit()
        image = cv2.imread(local_path)
        #print(image)
        #exit()
        images.append(image)

    correction1 = 0.1
    correction2 = 0.3
    #correction = 0.1  # this can be tried too
    #measurement = line[3]
    measurement = float(line[3])
    measurements.append(measurement)
    measurements.append(measurement + correction1)
    measurements.append(measurement - correction2)
#print(len(images))
#print(len(measurements))
#exit()
#print(images[0].shape)
#exit()
# adding augmented and flipped images to feed additional data into model and improve the
augmented_images = []
augmented_measurements = []
for image, measurement in zip(images, measurements):
    augmented_images.append(image)
    augmented_measurements.append(measurement)
    flipped_image = cv2.flip(image, 1)
    #flipped_measurement = float(measurement) *-1.0
    flipped_measurement = measurement *-1.0   #//looks this is better than above line
    augmented_images.append(flipped_image)
    augmented_measurements.append(flipped_measurement)
X_train = np.array(augmented_images)
y_train = np.array(augmented_measurements)
#print(len(y_train))
#print (X_train.shape)
#exit()
model = Sequential()
model.add(Lambda(lambda x: x /255.0-0.5, input_shape = (160, 320, 3)))   #2. added after
model.add(Cropping2D(cropping =((70,25),(0,0))))

model.add(Convolution2D(24,5,5, subsample =(2,2), activation = 'relu'))
model.add(Convolution2D(36,5,5, subsample =(2,2), activation = 'relu'))
model.add(Convolution2D(48,5,5, subsample =(2,2), activation = 'relu'))
model.add(Convolution2D(64,3,3, activation = 'relu'))
model.add(Convolution2D(64,3,3, activation = 'relu'))
model.add(Flatten())
#Adding a dropout layer to avoid overfitting by 50% rate
```

```
        model.add(Dropout(0.5))
        model.add(Dense(100))
        # Adding ELU
        #model.add(Activation('elu'))
        model.add(Dense(50))
        # Adding ELU
        #model.add(Activation('elu'))
        model.add(Dense(10))
        # Adding ELU
        #model.add(Activation('elu'))
        model.add(Dense(1))

        model.compile(optimizer = 'adam', loss = 'mse' )
        model.fit(X_train, y_train, validation_split =0.2, shuffle = True, epochs = 10)
        model.save('model.h5')
```

In [ ]:
```
        from moviepy.editor import ImageSequenceClip
        import argparse
        import os

        IMAGE_EXT = ['jpeg', 'gif', 'png', 'jpg']


        def main():
            parser = argparse.ArgumentParser(description='Create driving video.')
            parser.add_argument(
                'image_folder',
                type=str,
                default='',
                help='Path to image folder. The video will be created from these images.'
            )
            parser.add_argument(
                '--fps',
                type=int,
                default=60,
                help='FPS (Frames per second) setting for the video.')
            args = parser.parse_args()

            #convert file folder into list firltered for image file types
            image_list = sorted([os.path.join(args.image_folder, image_file)
                                 for image_file in os.listdir(args.image_folder)])

            image_list = [image_file for image_file in image_list if os.path.splitext(image_file

            #two methods of naming output video to handle varying environemnts
            video_file_1 = args.image_folder + '.mp4'
            video_file_2 = args.image_folder + 'output_video.mp4'
```

```
    print("Creating video {}, FPS={}".format(args.image_folder, args.fps))
    clip = ImageSequenceClip(image_list, fps=args.fps)

    try:
        clip.write_videofile(video_file_1)
    except:
        clip.write_videofile(video_file_2)


if __name__ == '__main__':
    main()
```