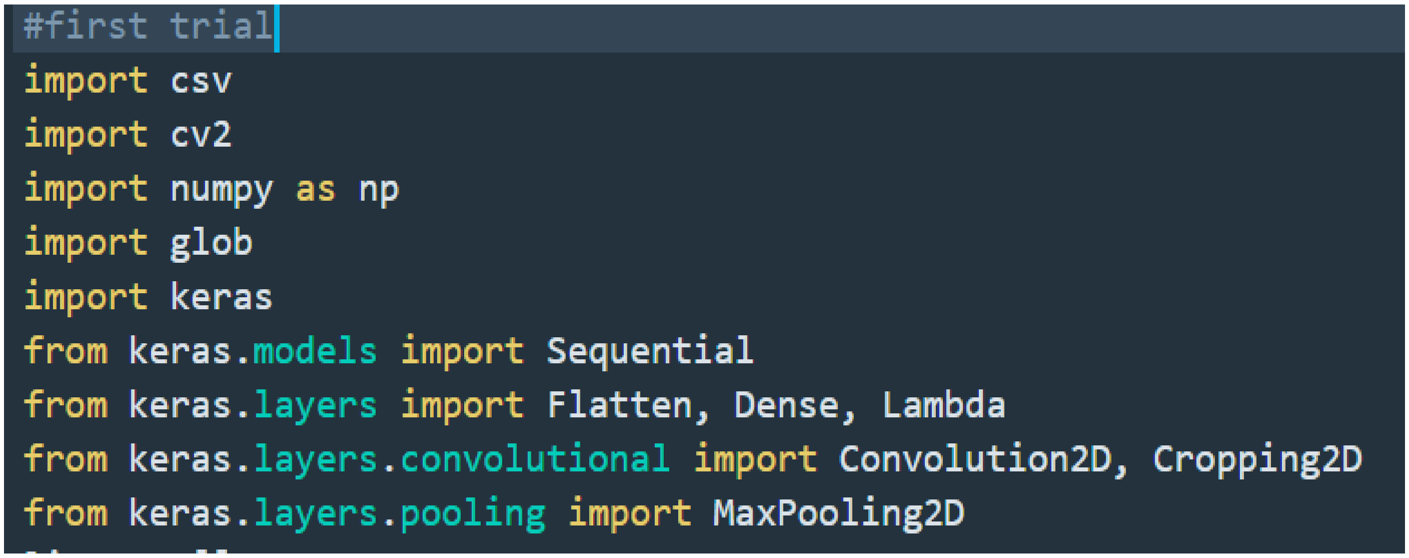
I started the project with going into simulator and drive the vehicle almost for two laps. Then I created a folder named "mydata" and imported the CSV file saved the data in the workspace and imported CSV file and images into it.

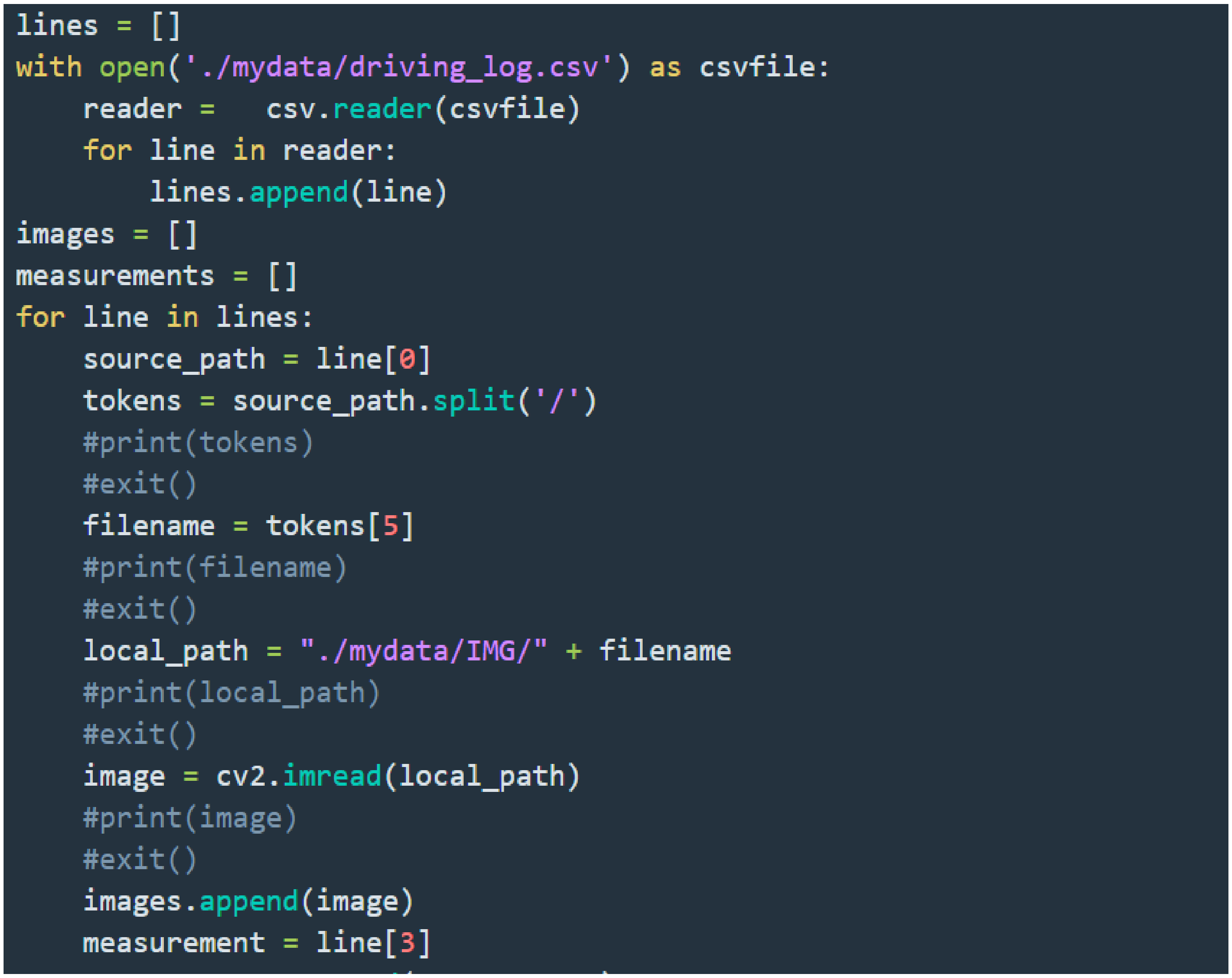
Although my final codes are in "model.py", I achieved my final code just after I did several trials and developed several ideas. Those ideas are shown in m1.py to m4.py files. You don't need to run all those files and they are there just for records of the trials I did. In summary, here are the sequence of steps I took:

1) **m1.py:** I imported libraries required and added them during writing my codes in several steps:



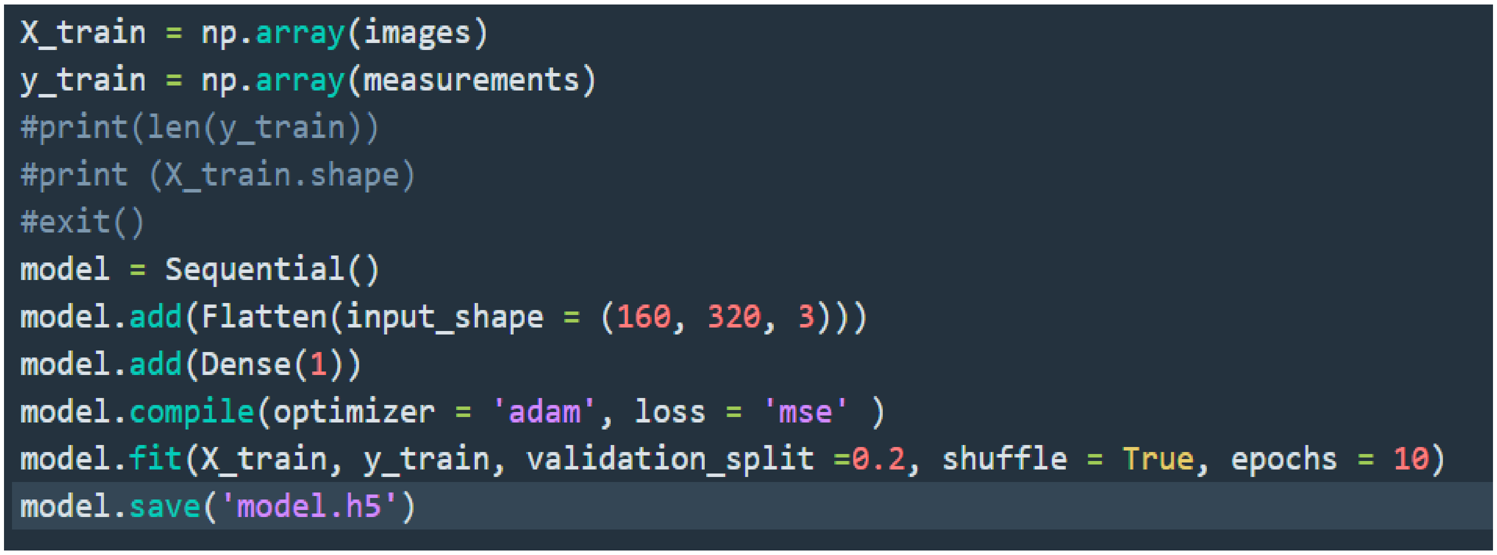
```
#first trial
import csv
import cv2
import numpy as np
import glob
import keras
from keras.models import Sequential
from keras.layers import Flatten, Dense, Lambda
from keras.layers.convolutional import Convolution2D, Cropping2D
from keras.layers.pooling import MaxPooling2D
```

Then I continued with data management, and imported the information from the CSV file into a matrix. Then I extracted the names of the center images one by one in a loop and found the equivalent image in the image directory (IMG) and appended them into a matrix. Also I read the measurement data and appended them in another matrix. For this I did test the codes step by step through several combination of print() and exit() commands:



```
lines = []
with open('./mydata/driving_log.csv') as csvfile:
    reader =   csv.reader(csvfile)
    for line in reader:
        lines.append(line)
images = []
measurements = []
for line in lines:
    source_path = line[0]
    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)
    measurement = line[3]
```

I then developed training data (X_train , y-train) from the images and measurements matrices. Then I used a simple sequential model with parameters shown below, and used adam optimizer and mse (Mean Square Error) for calculating the model loss. I used split range of .2 (80% data for training, and 20% for validation). I decided to use 10 epochs and finally I saved the model as “model.h5”:

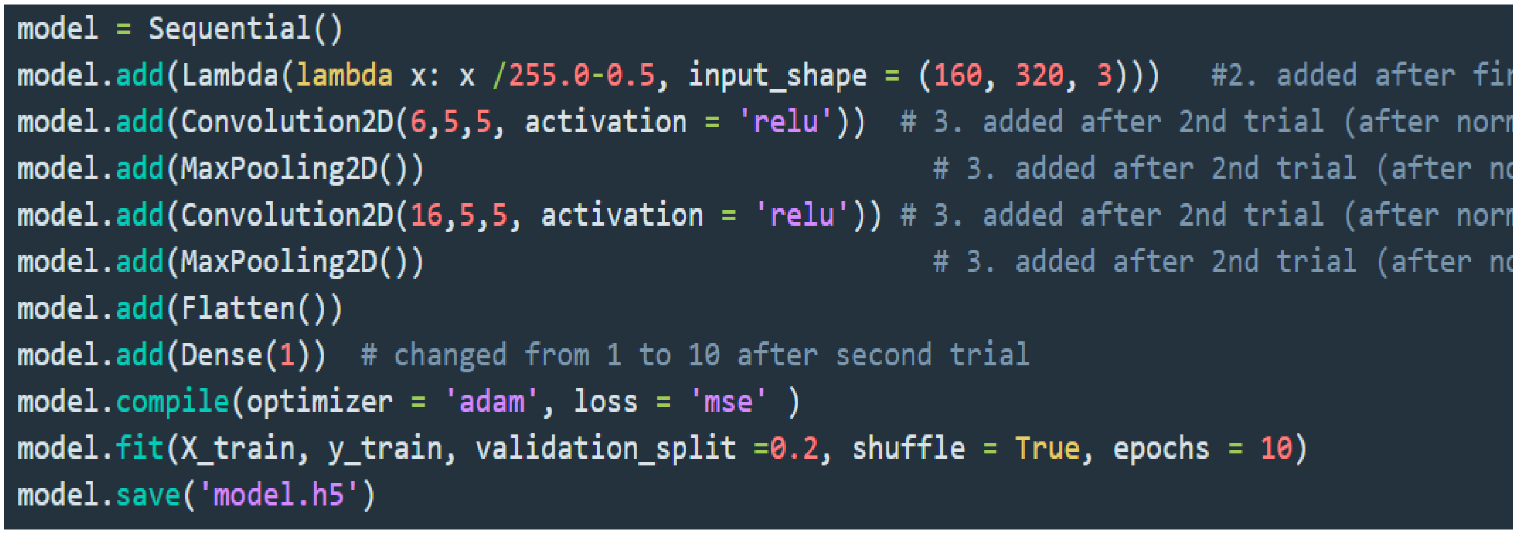


```
X_train = np.array(images)
y_train = np.array(measurements)
#print(len(y_train))
#print (X_train.shape)
#exit()
model = Sequential()
model.add(Flatten(input_shape = (160, 320, 3)))
model.add(Dense(1))
model.compile(optimizer = 'adam', loss = 'mse' )
model.fit(X_train, y_train, validation_split =0.2, shuffle = True, epochs = 10)
model.save('model.h5')
```

I the used command : “python m1.py” aand ran the codes. I noticed the amount of loss in each epochs is high.

I then ran the command : “python drive.py model.h5”, and then went into simulator and chose autonomuos mode and checked results: The vehicle started and after few seconds was droven to the left and went outside of the road.

2) **m2.py:** I decided to normalize the data by using Lambda for normalization. So I added it after introducing the sequential model into my code. I also decided to try LENET and CNN this time. I added 2 convolution and relu, and also added 2 Max poolings, and added codes as below:

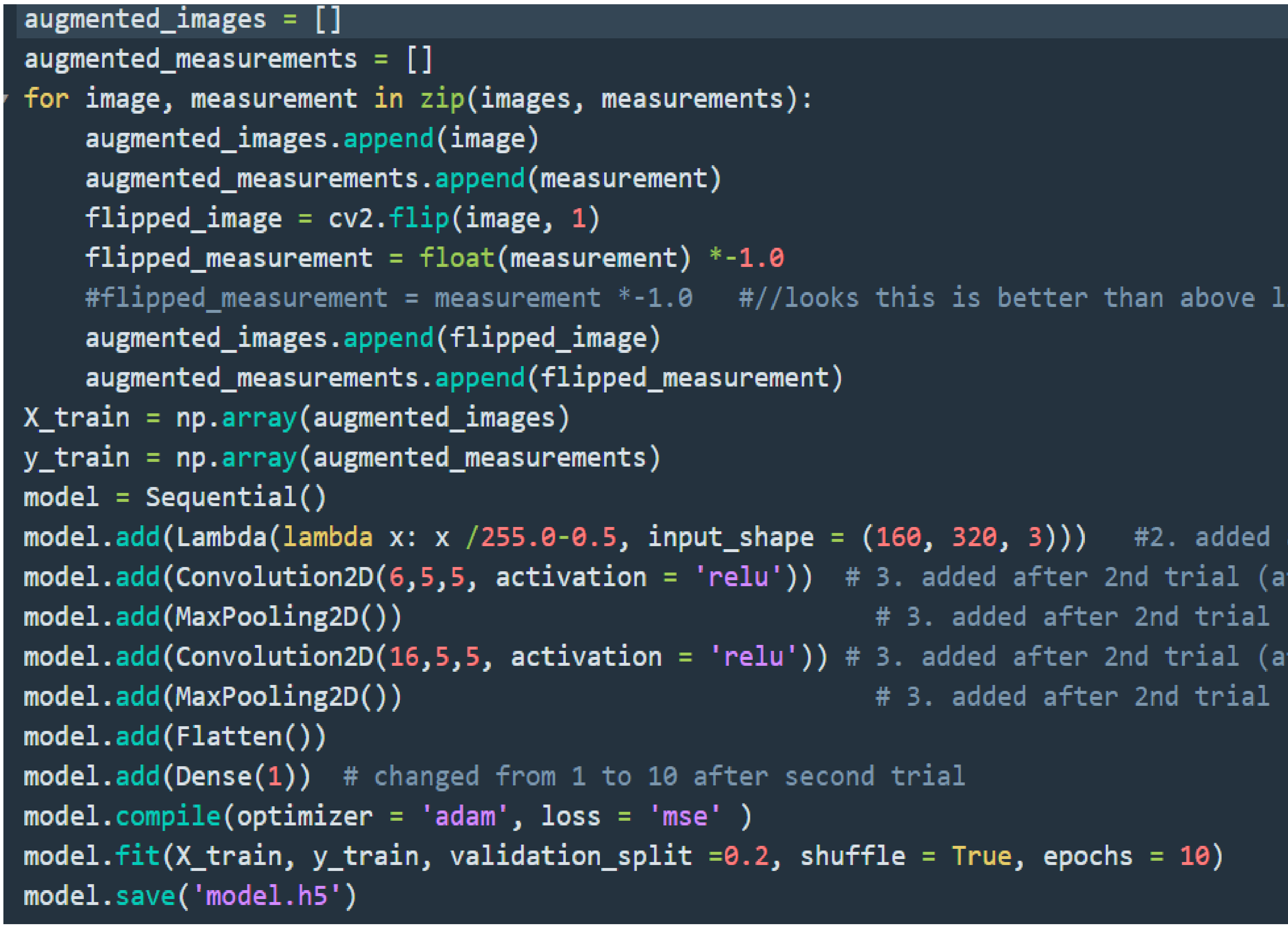


```
model = Sequential()
model.add(Lambda(lambda x: x /255.0-0.5, input_shape = (160, 320, 3)))   #2. added after fi
model.add(Convolution2D(6,5,5, activation = 'relu'))  # 3. added after 2nd trial (after nor
model.add(MaxPooling2D())                              # 3. added after 2nd trial (after n
model.add(Convolution2D(16,5,5, activation = 'relu')) # 3. added after 2nd trial (after nor
model.add(MaxPooling2D())                              # 3. added after 2nd trial (after n
model.add(Flatten())
model.add(Dense(1))  # changed from 1 to 10 after second trial
model.compile(optimizer = 'adam', loss = 'mse' )
model.fit(X_train, y_train, validation_split =0.2, shuffle = True, epochs = 10)
model.save('model.h5')
```

I did similar steps to run the code snd trin the model. I realized that loss has been decreased significantly in each epochs.

I then went to simulator again and checked the results: it was slightlyu better ,but again after few seconds (little bit longer than the first trial), it went towards left side of the road and went out and didn’t come back.

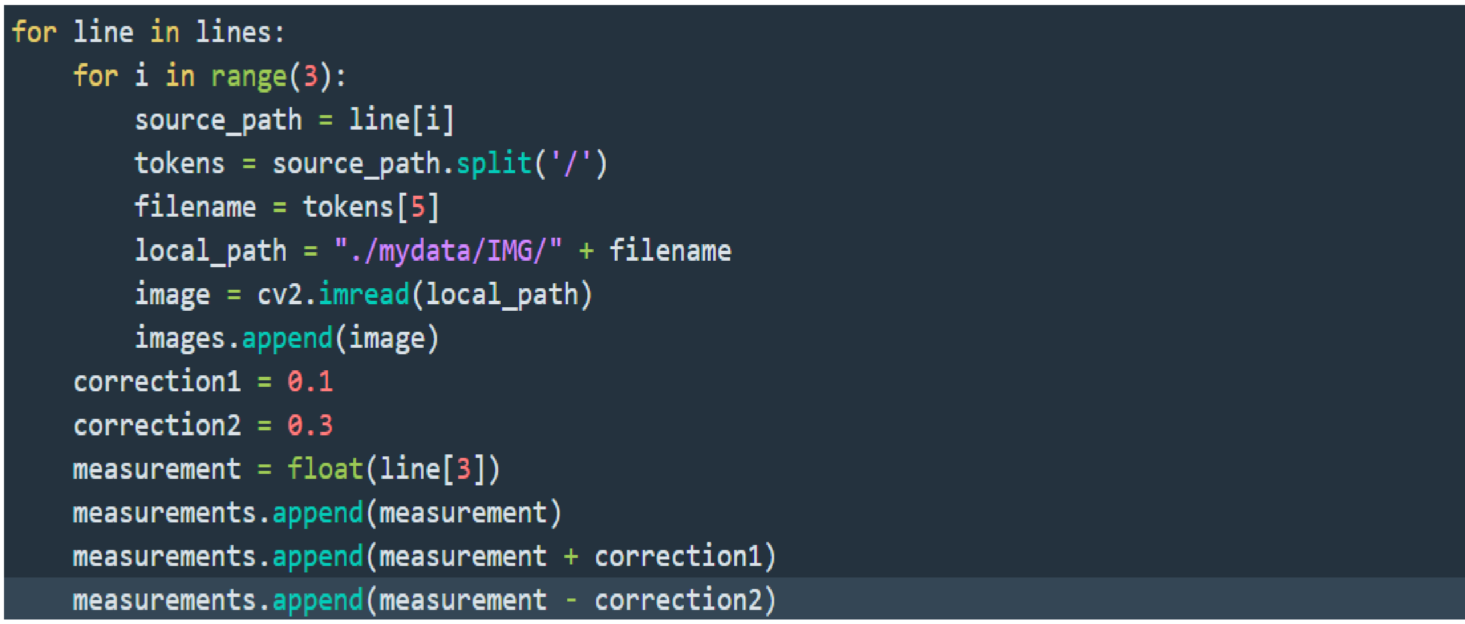
3) **m3.py:** I decided to generate some fake data by flipping the images to give the model more training data and to get better experience in dealing with situations where it slipes to the left. I also inversed the measurtements data for flipped images:



```
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 l
    augmented_images.append(flipped_image)
    augmented_measurements.append(flipped_measurement)
X_train = np.array(augmented_images)
y_train = np.array(augmented_measurements)
model = Sequential()
model.add(Lambda(lambda x: x /255.0-0.5, input_shape = (160, 320, 3)))   #2. added
model.add(Convolution2D(6,5,5, activation = 'relu'))  # 3. added after 2nd trial (a
model.add(MaxPooling2D())                               # 3. added after 2nd trial
model.add(Convolution2D(16,5,5, activation = 'relu')) # 3. added after 2nd trial (a
model.add(MaxPooling2D())                               # 3. added after 2nd trial
model.add(Flatten())
model.add(Dense(1))  # changed from 1 to 10 after second trial
model.compile(optimizer = 'adam', loss = 'mse' )
model.fit(X_train, y_train, validation_split =0.2, shuffle = True, epochs = 10)
model.save('model.h5')
```

Then I did same steps for training the model and then watching the results in the simulator. It imporved significantly however in half the way before the bridge slipped to the right this time and went out of the road.

4) **m4.py:** I then decided to try the idea of using left and right images from left and right cameras too. I also decided to crop the image from top and bototm and delete usless data fro meach image as (around 70 pixels from top and 25 pixels from bottom of each image). For that I also defined two parrameters to generate fake data for mesurements (steering) and added and subtracted the mfrom the real measurements data: correction1 and correction2.

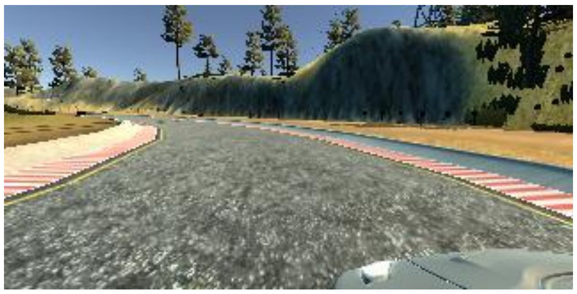


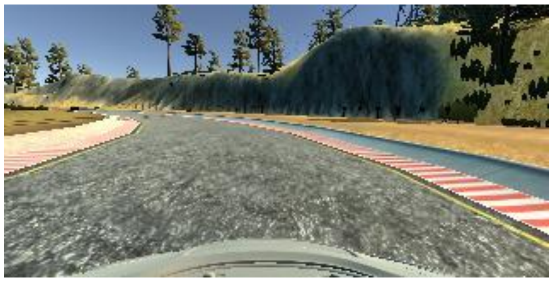
```python
for line in lines:
    for i in range(3):
        source_path = line[i]
        tokens = source_path.split('/')
        filename = tokens[5]
        local_path = "./mydata/IMG/" + filename
        image = cv2.imread(local_path)
        images.append(image)
    correction1 = 0.1
    correction2 = 0.3
    measurement = float(line[3])
    measurements.append(measurement)
    measurements.append(measurement + correction1)
    measurements.append(measurement - correction2)
```

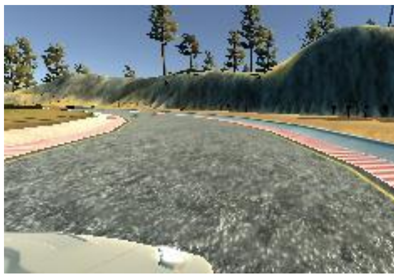
Then I did four trials by four sets of correction parameters:

- correction1 = correction2= 0.2
- correction1 = correction2= 0.1
- correction1 = 0.3, correction2= 0.1
- correction1 = 0.1, correction2= 0.3

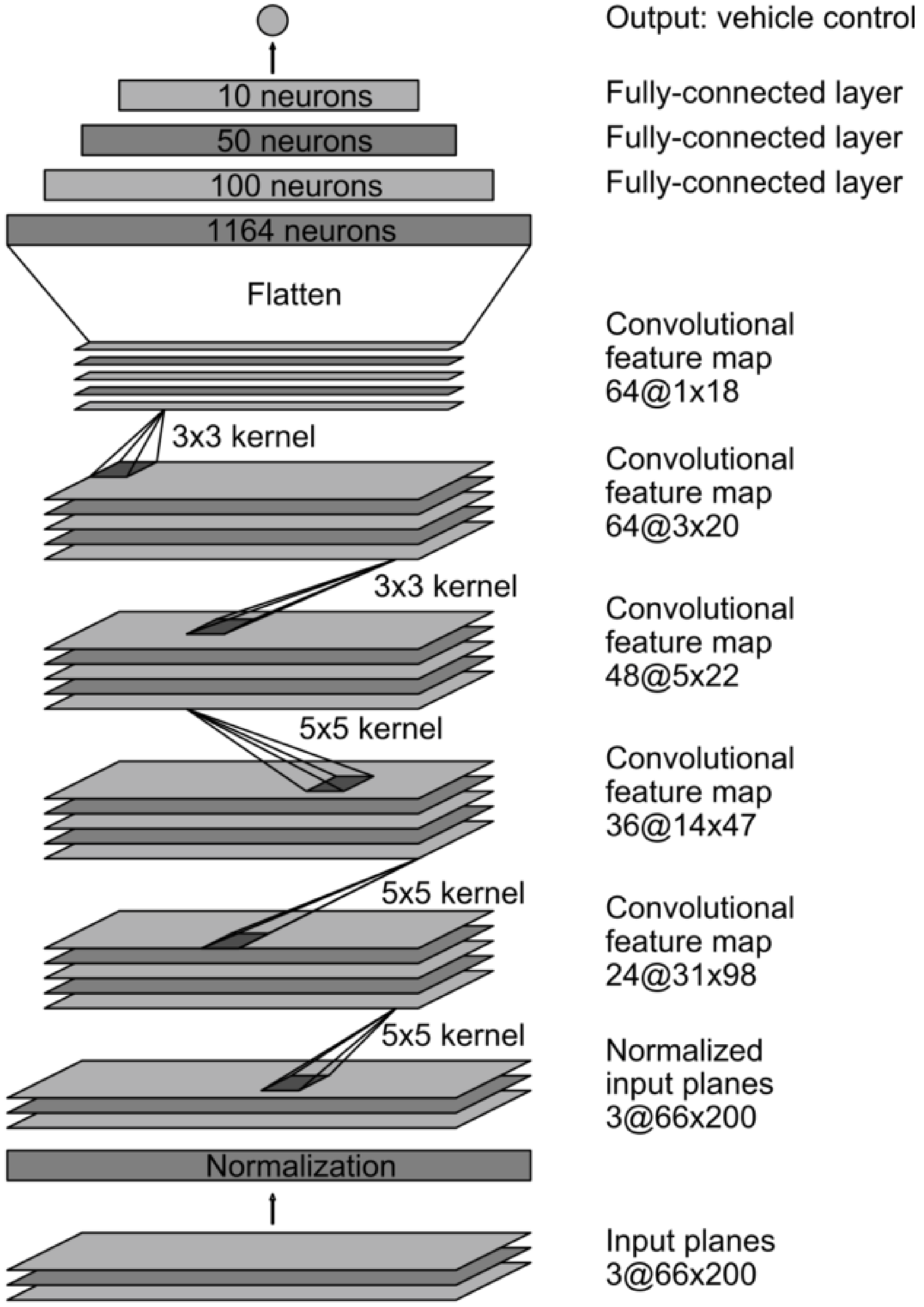
As far as I remember, in the first trial, the slided to right and exited the road in the curve before the bridge. For the other 3 trials it passed the bridge and slided to the right in the first curve after the bridge. Overal I noticed little of improvement in case 4 (correction1 = 0.1, correction2= 0.3). Below are examples of left, center, and right images (from left, center, and right cameras):


Left

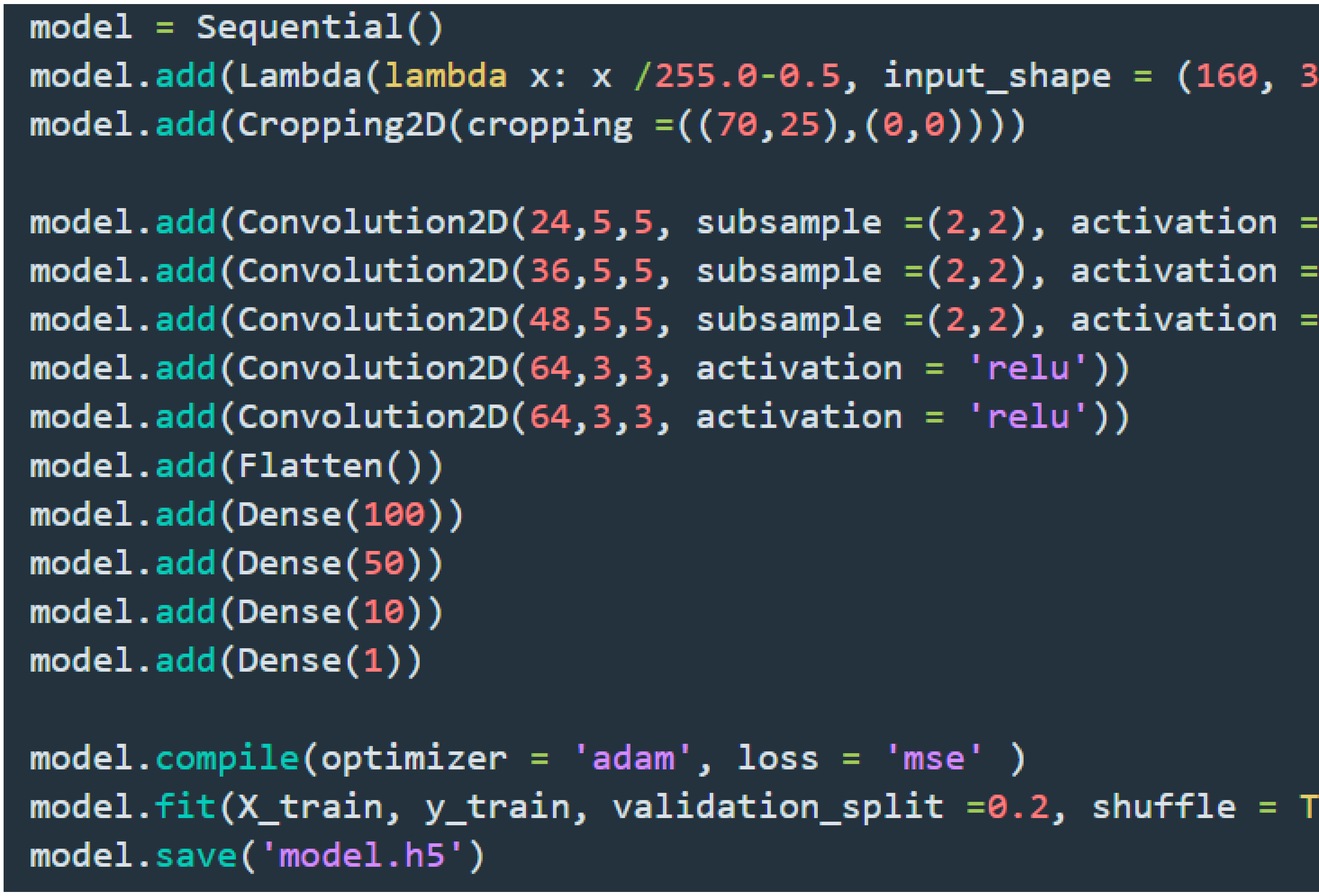

Center


Right

5) **model.py:** I decided to try nvidia end-to-end learning for autonomuos driving model shown below:



For that and based of above image, I developed below code additionally:



```
model = Sequential()
model.add(Lambda(lambda x: x /255.0-0.5, input_shape = (160, 3
model.add(Cropping2D(cropping =((70,25),(0,0))))

model.add(Convolution2D(24,5,5, subsample =(2,2), activation =
model.add(Convolution2D(36,5,5, subsample =(2,2), activation =
model.add(Convolution2D(48,5,5, subsample =(2,2), activation =
model.add(Convolution2D(64,3,3, activation = 'relu'))
model.add(Convolution2D(64,3,3, activation = 'relu'))
model.add(Flatten())
model.add(Dense(100))
model.add(Dense(50))
model.add(Dense(10))
model.add(Dense(1))

model.compile(optimizer = 'adam', loss = 'mse' )
model.fit(X_train, y_train, validation_split =0.2, shuffle = T
model.save('model.h5')
```

Then I did the same steps to train, validate, and then see the results in simulator. YThis time the vehicled drived nicely and beautifully all the lap.