SDC Chp10

July 14, 2022

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from keras import layers
from keras.models import Sequential
from keras.optimizers import Adam
from keras.layers import Convolution2D, MaxPooling2D, Dropout, Flatten, Dense

import cv2
import pandas as pd
import random
import os
import ntpath
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
import matplotlib.image as mpimg
from imgaug import augmenters as iaa
```

1 Dat preparation

[ ]:
```
datadir = "Data_Collect"
columns = ["center", "left", "right", "steering", "throttle",
"reverse", "speed"]

#data = pd.read_csv(os.path.join(datadir, "driving_log.csv"), names = columns)
data = pd.read_csv(os.path.join(datadir, "driving_log.csv"), names = columns)
pd.set_option("display.max_colwidth", -1)

#data.head()
```

```
/var/folders/wj/2bdfjxmx6_dd5gvnsyg6_rvc0000gn/T/ipykernel_3423/4251334168.py:7:
FutureWarning: Passing a negative integer is deprecated in version 1.0 and will
not be supported in future version. Instead, use None to not limit the column
width.
  pd.set_option("display.max_colwidth", -1)
```

[ ]:
```
def path_leaf(path):
    head, tail = ntpath.split(path)
    return tail

data['center'] = data['center'].apply(path_leaf)
data['left'] = data['left'].apply(path_leaf)
data['right'] = data['right'].apply(path_leaf)
data.head()
```

[ ]:

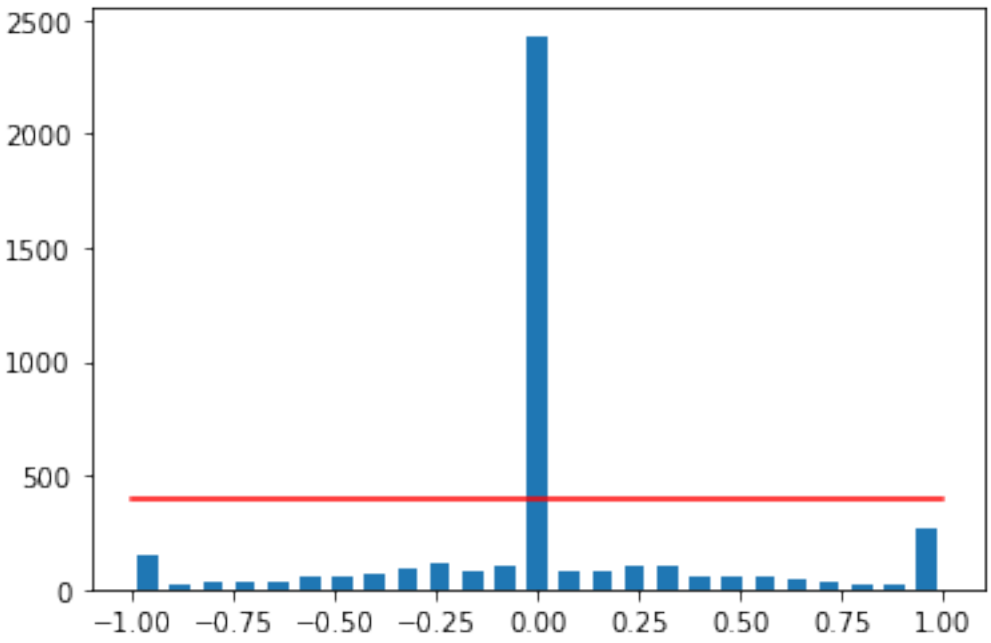
	center	left	right	steering	throttle	reverse	speed
0	center_2022_07_12_22_56_41_414.jpg	left_2022_07_12_22_56_41_414.jpg	right_2022_07_12_22_56_41_414.jpg	0.0	0.0	0.0	0.000132
1	center_2022_07_12_22_56_41_530.jpg	left_2022_07_12_22_56_41_530.jpg	right_2022_07_12_22_56_41_530.jpg	0.0	0.0	0.0	0.000141
2	center_2022_07_12_22_56_41_647.jpg	left_2022_07_12_22_56_41_647.jpg	right_2022_07_12_22_56_41_647.jpg	0.0	0.0	0.0	0.000130
3	center_2022_07_12_22_56_41_765.jpg	left_2022_07_12_22_56_41_765.jpg	right_2022_07_12_22_56_41_765.jpg	0.0	0.0	0.0	0.000153
4	center_2022_07_12_22_56_41_882.jpg	left_2022_07_12_22_56_41_882.jpg	right_2022_07_12_22_56_41_882.jpg	0.0	0.0	0.0	0.000261

1.1 1) Steering angle visualization

1.1.1 We can see that most of the data have steering angle data = 0

[ ]:
```
num_bins = 25
samples_per_bin = 400

hist, bins = np.histogram(data["steering"], num_bins)

center = (bins[:-1] + bins[1:]) * 0.5
plt.bar(center, hist, width=0.05)
plt.plot((np.min(data["steering"]), np.max(data["steering"])),␣
 ↪(samples_per_bin, samples_per_bin), color = "red")
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x2cd37bd00>]
```



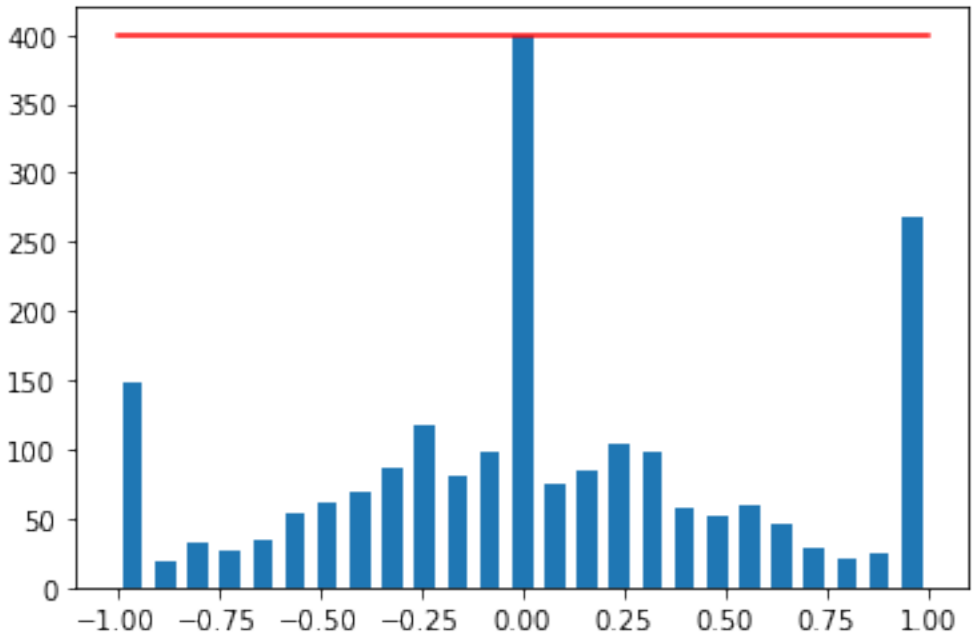
1.1.2 Reduce every bin to only have 400 data. After reduce the number of data, we can see that each bin have max 400 data points

[ ]:
```
#for j in range(num_bins):
#    print(bins[j])
#print("--------")


from turtle import color


print('total data:', len(data))
remove_list = []
for j in range(num_bins):
    #print(bins[j])
    list_ = []
    for i in range(len(data['steering'])):
        # Check each bins only have 400 data
        if data['steering'][i] >= bins[j] and data['steering'][i] <= bins[j+1]:

            #print(data['steering'][i])
            list_.append(i)
    list_ = shuffle(list_)
    #set the number of data in bins to 400
    list_ = list_[samples_per_bin:]
```

```
    remove_list.extend(list_)

print('removed:', len(remove_list))
# Remove the data with steering angle = 0 to max number 400
data.drop(data.index[remove_list], inplace=True)
print('remaining:', len(data))

hist, _ = np.histogram(data['steering'], (num_bins))
plt.bar(center, hist, width=0.05)
plt.plot((np.min(data['steering']), np.max(data['steering'])),␣
 ↪(samples_per_bin, samples_per_bin), color = "red")
```

```
total data: 4179
removed: 2030
remaining: 2149
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x2d0c131f0>]
```



1.2 2) Extract the image's path and the corresponding steering angle information

[ ]:
```
print(data.iloc[1])
print("----------")
def load_img_steering(datadir, df):
    image_path = []
    steering = []
```

```
    for i in range(len(data)):
        indexed_data = data.iloc[i]
        center, left, right = indexed_data[0], indexed_data[1], indexed_data[2]
        image_path.append(os.path.join(datadir, center.strip()))
        steering.append(float(indexed_data[3]))
        # left image append
        image_path.append(os.path.join(datadir,left.strip()))
        steering.append(float(indexed_data[3])+0.15)
        # right image append
        image_path.append(os.path.join(datadir,right.strip()))
        steering.append(float(indexed_data[3])-0.15)
    image_paths = np.asarray(image_path)
    steerings = np.asarray(steering)
    return image_paths, steerings

image_paths, steerings = load_img_steering(datadir + '/IMG', data)
#print(image_paths)
```

```
center      center_2022_07_13_21_08_26_974.jpg
left        left_2022_07_13_21_08_26_974.jpg
right       right_2022_07_13_21_08_26_974.jpg
steering    0.35754
throttle    0.0
reverse     1.0
speed       1.326189
Name: 35, dtype: object
----------
```

1.3 3) Split the data using the sklearn library

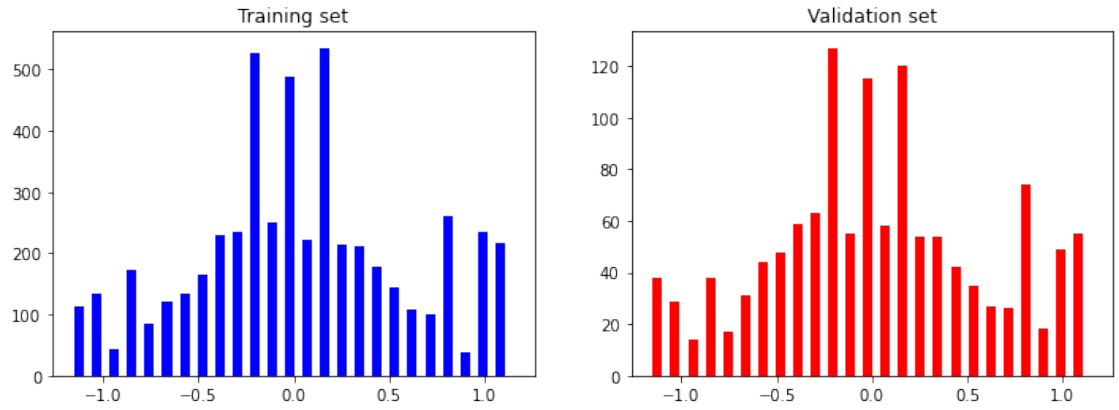
[ ]:
```
X_train, X_valid, y_train, y_valid = train_test_split(image_paths, steerings,␣
↪test_size=0.2, random_state=6)

print("Training samples", len(X_train))
print('Valid samples', len(X_valid))
```

```
Training samples 5157
Valid samples 1290
```

1.3.1 Check the distribution of the training and test dataset

[ ]:
```
fig, axes = plt.subplots(1, 2, figsize = (12, 4))
axes[0].hist(y_train, bins = num_bins, width = 0.05, color = "blue")
axes[0].set_title("Training set")

axes[1].hist(y_valid, bins = num_bins, width = 0.05, color = "red")
axes[1].set_title("Validation set")
```

[ ]: Text(0.5, 1.0, 'Validation set')



1.4 4) Explore and clean the data.

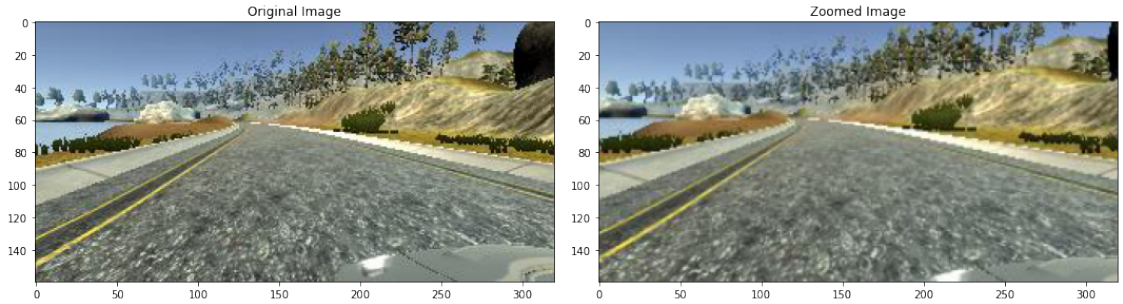
1.4.1 Applied various computer vision filtering echniques to it.

1.4.2 4.1) Zooming in on the images

[ ]:
```
def zoom(image):
    #Affine transformations involve Scaling ("zoom" in/out)
    zoom = iaa.Affine(scale = (1, 1.3))
    image = zoom.augment_image(image)
    return image
```

[ ]:
```
image = image_paths[random.randint(0, 1000)]
original_image = mpimg.imread(image)
zoomed_image = zoom(original_image)

fig, axs = plt.subplots(1, 2, figsize =(15, 10))
fig.tight_layout()

axs[0].imshow(original_image)
axs[0].set_title("Original Image")

axs[1].imshow(zoomed_image)
axs[1].set_title("Zoomed Image")
```

[ ]: Text(0.5, 1.0, 'Zoomed Image')

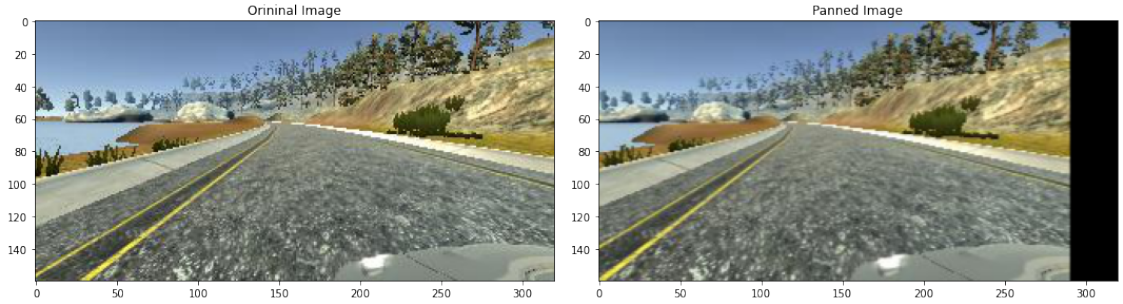


1.4.3 4.2) Pan the images

[ ]:
```
def pan(image):
    #Translation ("move" image on the x-/y-axis)
    pan = iaa.Affine(translate_percent= {"x" : (-0.1, 0.1), "y": (-0.1, 0.1)})
    image = pan.augment_image(image)
    return image
```

[ ]:
```
image = image_paths[random.randint(0, 1000)]
original_image = mpimg.imread(image)
panned_image = pan(original_image)

fig, axs = plt.subplots(1, 2, figsize=(15, 10))
fig.tight_layout()

axs[0].imshow(original_image)
axs[0].set_title("Orininal Image")

axs[1].imshow(panned_image)
axs[1].set_title("Panned Image")
```

[ ]:
```
Text(0.5, 1.0, 'Panned Image')
```

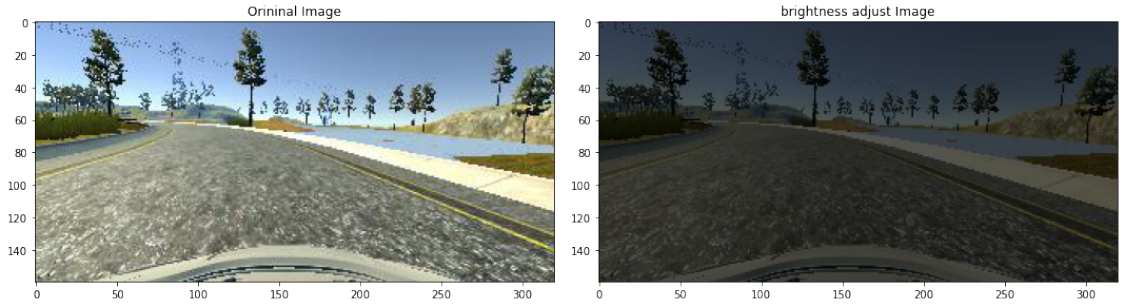


1.4.4 4.3) Adjust the brightness of the images

[ ]:
```
def img_random_brightness(image):
    brigthness = iaa.Multiply((0.2, 1.2))
    image = brigthness.augment_image(image)
    return image
```

[ ]:
```
image = image_paths[random.randint(0, 1000)]
original_image = mpimg.imread(image)
bright_image = img_random_brightness(original_image)

fig, axs = plt.subplots(1, 2, figsize=(15, 10))
fig.tight_layout()

axs[0].imshow(original_image)
axs[0].set_title("Orininal Image")

axs[1].imshow(bright_image)
axs[1].set_title("brightness adjust Image")
```

[ ]:
```
Text(0.5, 1.0, 'brightness adjust Image')
```



1.4.5 4.4) Flip the images horizontally and vertically

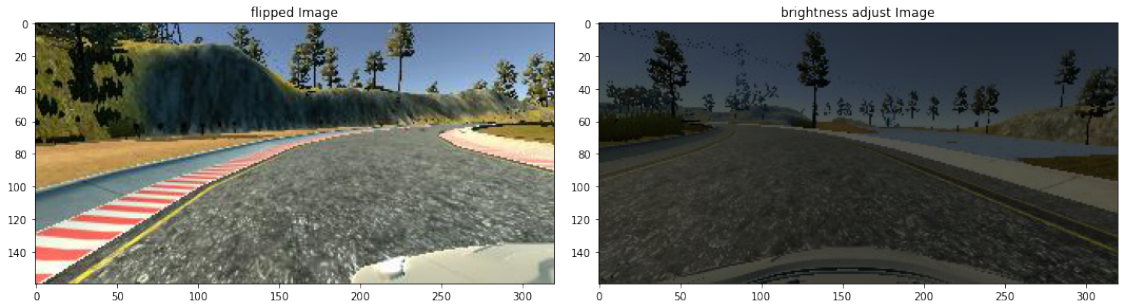
[ ]:
```
def img_random_flip(image, steering_angle):
    image = cv2.flip(image, 1)
    steering_angle = -steering_angle
    return image, steering_angle
```

[ ]:
```
random_index = random.randint(0, 1000)
print(random_index)
image = image_paths[random_index]
steering_angle = steerings[random_index]


original_image = mpimg.imread(image)
```

```
flipped_image, flipped_steering_angle = img_random_flip(original_image,␣
 ↪steering_angle = steering_angle)

fig, axs = plt.subplots(1, 2, figsize=(15, 10))
fig.tight_layout()

axs[0].imshow(flipped_image)
axs[0].set_title("flipped Image")

axs[1].imshow(bright_image)
axs[1].set_title("brightness adjust Image")
```

617

[ ]: Text(0.5, 1.0, 'brightness adjust Image')

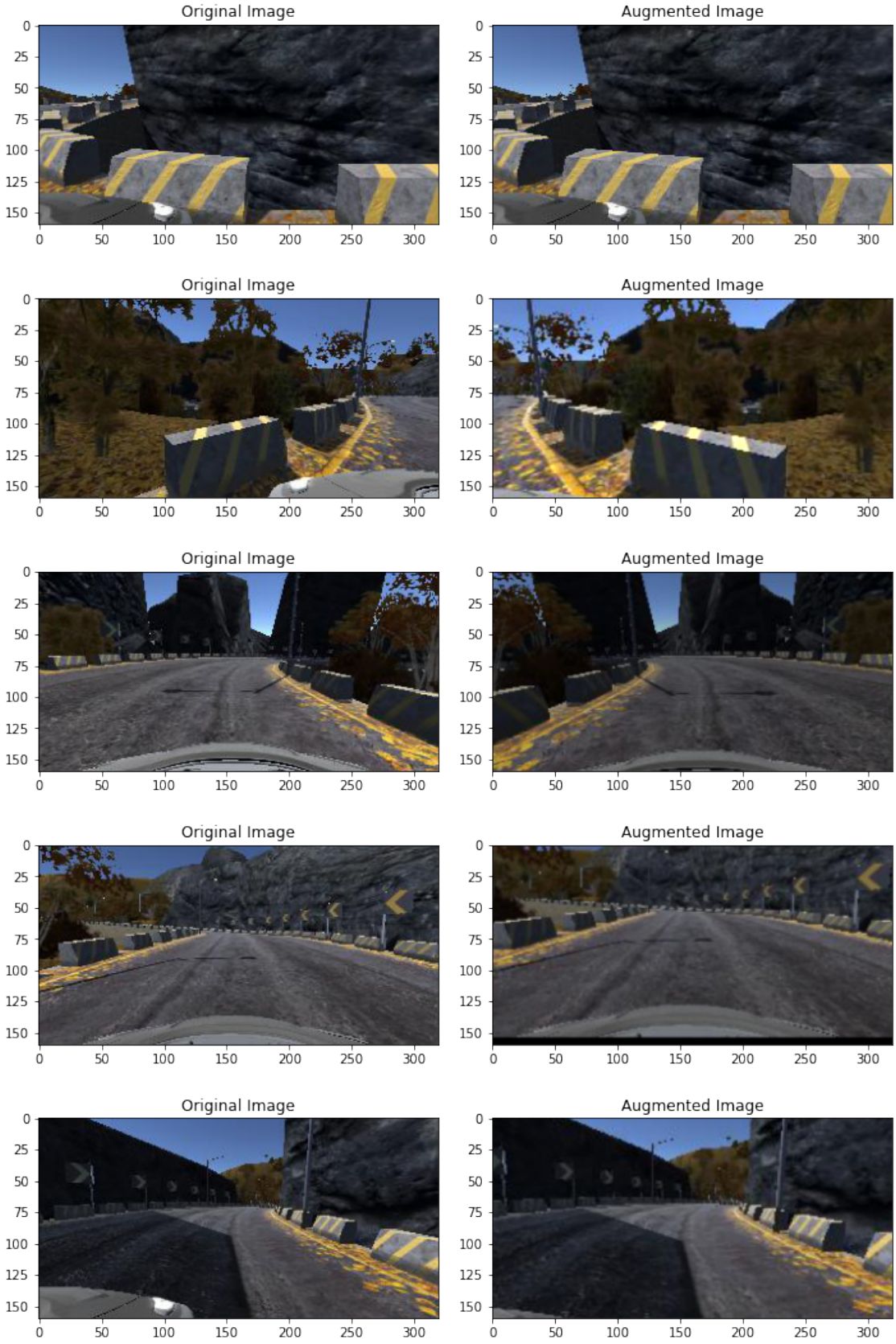


1.5 5) Randomly applied filter on images

[ ]:
```
def random_augment(image, steering_angle):
    image = mpimg.imread(image)
    if np.random.rand() < 0.5:
        image = pan(image)
    if np.random.rand() < 0.5:
        image = zoom(image)
    if np.random.rand() < 0.5:
        image = img_random_brightness(image)
    if np.random.rand() < 0.5:
        image, steering_angle = img_random_flip(image, steering_angle)

    return image, steering_angle
```

[ ]:
```
ncol = 2
nrow = 5

fig, axs = plt.subplots(nrow, ncol, figsize = (10, 15))
```

```
fig.tight_layout()

for i in range(5):
    randnum = random.randint(0, len(image_paths) - 1)
    random_image = image_paths[randnum]
    random_steering = steerings[randnum]

    original_image = mpimg.imread(random_image)
    augmented_image, steering = random_augment(random_image, random_steering)


    axs[i][0].imshow(original_image)
    axs[i][0].set_title("Original Image")

    axs[i][1].imshow(augmented_image)
    axs[i][1].set_title("Augmented Image")
```

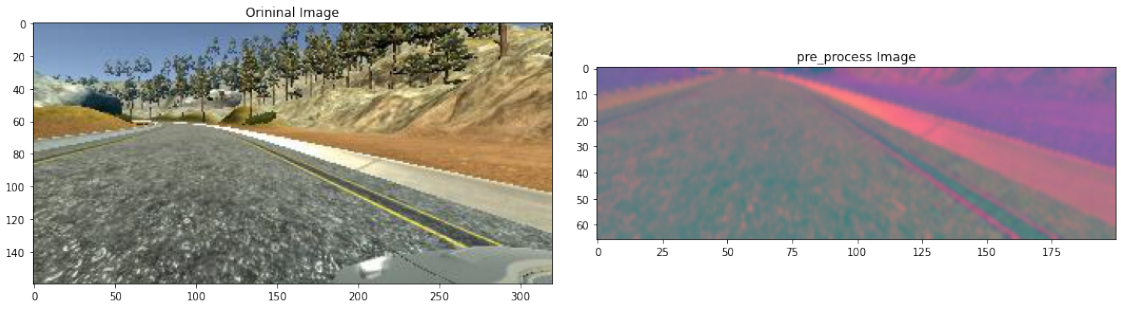

Original Image
Augmented Image
Original Image
Augmented Image
Original Image
Augmented Image
Original Image
Augmented Image
Original Image
Augmented Image

1.6 6) Cropping the image and applying Gaussian blur

[ ]:
```
#applied Gaussian blur
def img_preprocess(img):
    img = img[60:135, :, :]
    img = cv2.cvtColor(img, cv2.COLOR_RGB2YUV)
    img = cv2.GaussianBlur(img, (3,3), 0)
    img = cv2.resize(img, (200, 66))
    img = img / 255
    return img
```

[ ]:
```
image = image_paths[random.randint(0, 1000)]
original_image = mpimg.imread(image)
pre_process_image = img_preprocess(original_image)

fig, axs = plt.subplots(1, 2, figsize=(15, 10))
fig.tight_layout()

axs[0].imshow(original_image)
axs[0].set_title("Orininal Image")

axs[1].imshow(pre_process_image)
axs[1].set_title("pre_process Image")
```

[ ]:
```
Text(0.5, 1.0, 'pre_process Image')
```

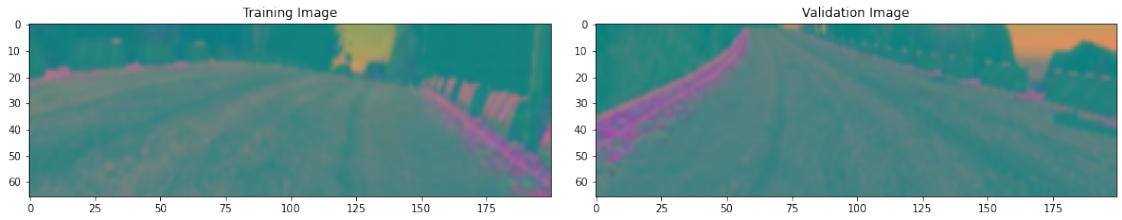


[ ]:
```
def batch_generator(image_paths, steering_ang, batch_size, istraining):

    while True:
        # : Create empty list for batch_img and batch_streeing for each batch processing
        batch_img = []
```

```
        batch_steering = []

        for i in range(batch_size):
            # : size-shaped array of random integers from the appropriate␣
↪distribution, or a single such random int if size not provided.
            random_index = random.randint(0, len(image_paths) - 1)
            # : Check batch-generator is training or testing.
            # : If it is training it performs random_aumentation method  and␣
↪get the img and steering angle
            # : else it reads the image paths with random index to get the␣
↪image  and steering angle
            if istraining:
                im, steering = random_augment(image_paths[random_index],␣
↪steering_ang[random_index])
            else:
                im = mpimg.imread(image_paths[random_index])
                steering = steering_ang[random_index]
            # : Perform Image processing techniques for converting a image to␣
↪YUV color, removing blur and resize the image
            im = img_preprocess(im)
            batch_img.append(im)
            batch_steering.append(steering)
        yield (np.asarray(batch_img), np.asarray(batch_steering))
```

[ ]:
```
x_train_gen, y_train_gen = next(batch_generator(X_train, y_train, 1, 1))
x_valid_gen, y_valid_gen = next(batch_generator(X_valid, y_valid, 1, 0))

fig, axs = plt.subplots(1, 2, figsize=(15, 10))
fig.tight_layout()

axs[0].imshow(x_train_gen[0])
axs[0].set_title('Training Image')

axs[1].imshow(x_valid_gen[0])
axs[1].set_title('Validation Image')
```

[ ]:
```
Text(0.5, 1.0, 'Validation Image')
```



2 Model development

[ ]:

[ ]: