

```
!git clone https://github.com/pranit570/multiobstacle.git
```

```
↳ fatal: destination path 'multiobstacle' already exists and is not an empty directory.
```

```
!ls multiobstacle
```

```
↳ driving_log.csv  IMG
```

```
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import keras
from keras.models import Sequential
from keras.optimizers import Adam
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from imgaug import augmenters as iaa
import cv2
import pandas as pd
import ntpath
import random

datadir = 'multiobstacle'
columns = ['center', 'left', 'right', 'steering', 'throttle', 'reverse', 'speed']
data = pd.read_csv(os.path.join(datadir, 'driving_log.csv'), names = columns)
pd.set_option('display.max_colwidth', -1)
data.head()
```

```
↳ center
0 C:\1jan2019pranitlenovo\data\mtechparul\final year dissertation\data\9oct19\multiobstacle\IMG\center_2019_10_09_04_40_46_944.jpg dissertation\data\9oct19\mu
1 C:\1jan2019pranitlenovo\data\mtechparul\final year dissertation\data\9oct19\multiobstacle\IMG\center_2019_10_09_04_40_47_026.jpg dissertation\data\9oct19\mu
2 C:\1jan2019pranitlenovo\data\mtechparul\final year dissertation\data\9oct19\multiobstacle\IMG\center_2019_10_09_04_40_47_099.jpg dissertation\data\9oct19\mu
3 C:\1jan2019pranitlenovo\data\mtechparul\final year dissertation\data\9oct19\multiobstacle\IMG\center_2019_10_09_04_40_47_168.jpg dissertation\data\9oct19\mu
4 C:\1jan2019pranitlenovo\data\mtechparul\final year dissertation\data\9oct19\multiobstacle\IMG\center_2019_10_09_04_40_47_236.jpg dissertation\data\9oct19\mu
```

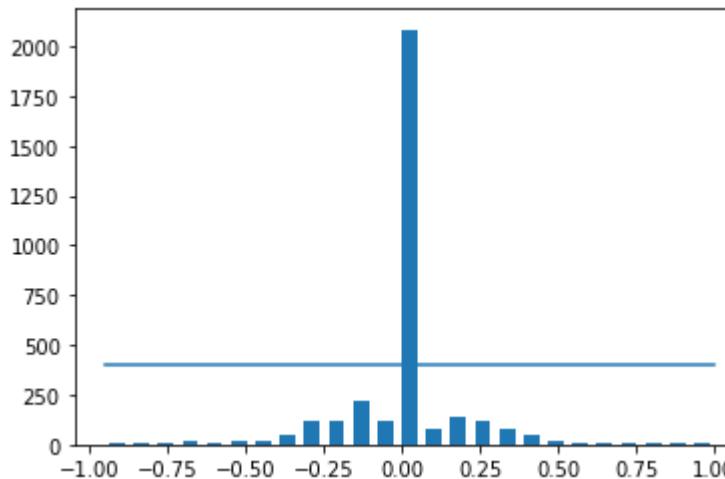
```
def path_leaf(path):
    head, tail = ntpath.split(path)
    return tail
data['center'] = data['center'].apply(path_leaf)
data['left'] = data['left'].apply(path_leaf)
data['right'] = data['right'].apply(path_leaf)
data.head()
num_bins = 25
```

```

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))

```

↳ [matplotlib.lines.Line2D at 0x7f2b2de43128]



```

print('total data:', len(data))
remove_list = []
for j in range(num_bins):
    list_ = []
    for i in range(len(data['steering'])):
        if data['steering'][i] >= bins[j] and data['steering'][i] <= bins[j+1]:
            list_.append(i)
    list_ = shuffle(list_)
    list_ = list_[samples_per_bin:]
    remove_list.extend(list_)

print('removed:', len(remove_list))
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))

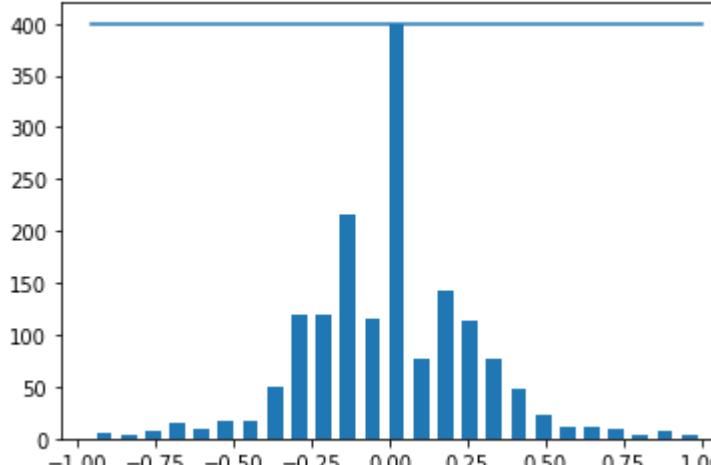
```

↳ total data: 3310

removed: 1683

remaining: 1627

[matplotlib.lines.Line2D at 0x7f2b28467e80]



```

print(data.iloc[1])
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)
```

```

↳ center      center_2019_10_09_04_40_47_236.jpg
    left       left_2019_10_09_04_40_47_236.jpg
    right      right_2019_10_09_04_40_47_236.jpg
    steering    0
    throttle   0
    reverse    0
    speed      5.18e-07
Name: 4, dtype: object

```

```
X_train, X_valid, y_train, y_valid = train_test_split(image_paths, steerings, test_size=0.2, random_
print('Training Samples: {} \nValid Samples: {}'.format(len(X_train), len(X_valid)))
```

```

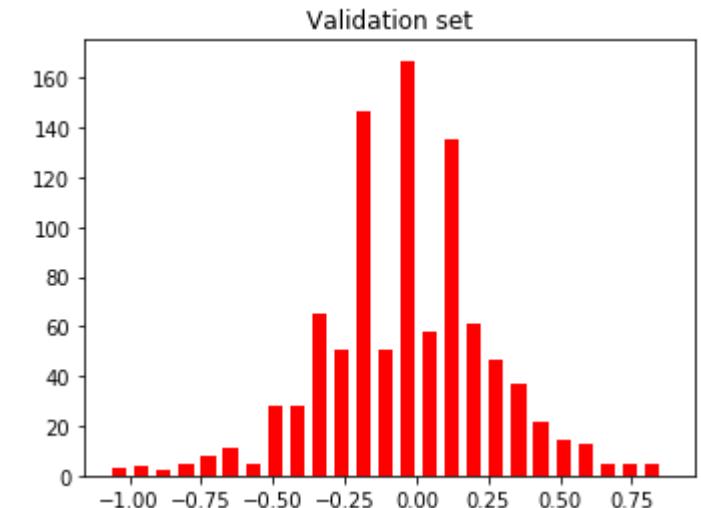
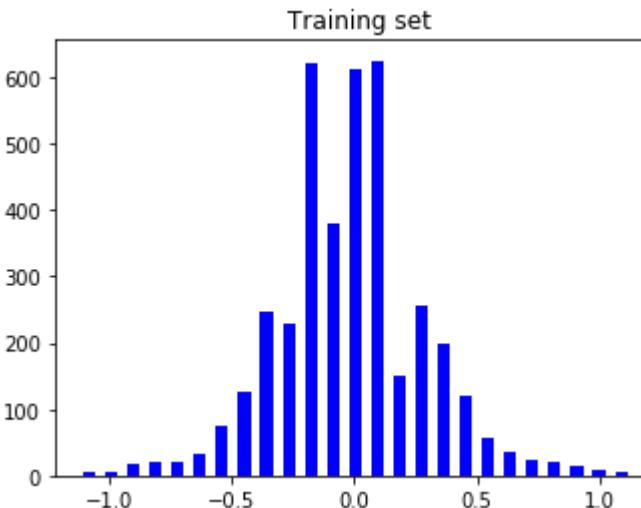
fig, axes = plt.subplots(1, 2, figsize=(12, 4))
axes[0].hist(y_train, bins=num_bins, width=0.05, color='blue')
axes[0].set_title('Training set')
axes[1].hist(y_valid, bins=num_bins, width=0.05, color='red')
axes[1].set_title('Validation set')

```

```

↳ Training Samples: 3904
    Valid Samples: 977
    Text(0.5, 1.0, 'Validation set')

```



```

def zoom(image):
    image = cv2.resize(image, (1, 1.2))

```

```

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)

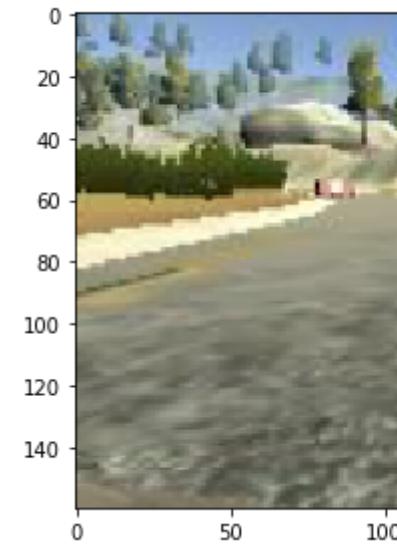
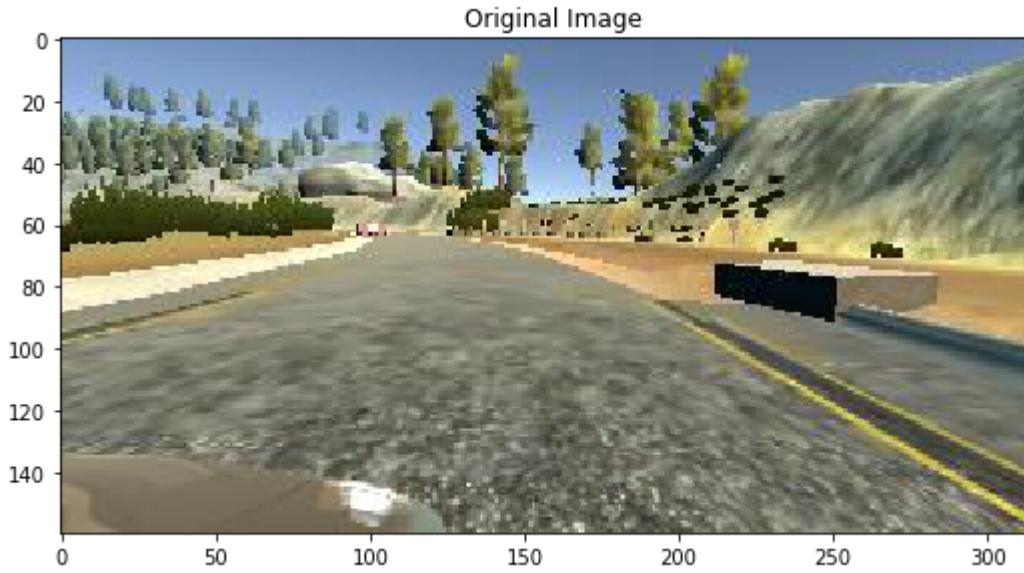
fig, axs = plt.subplots(1, 2, figsize=(15, 10))
fig.tight_layout()

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, 'Zoomed Image')



```

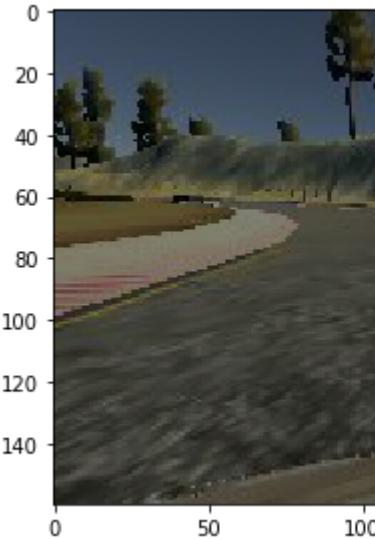
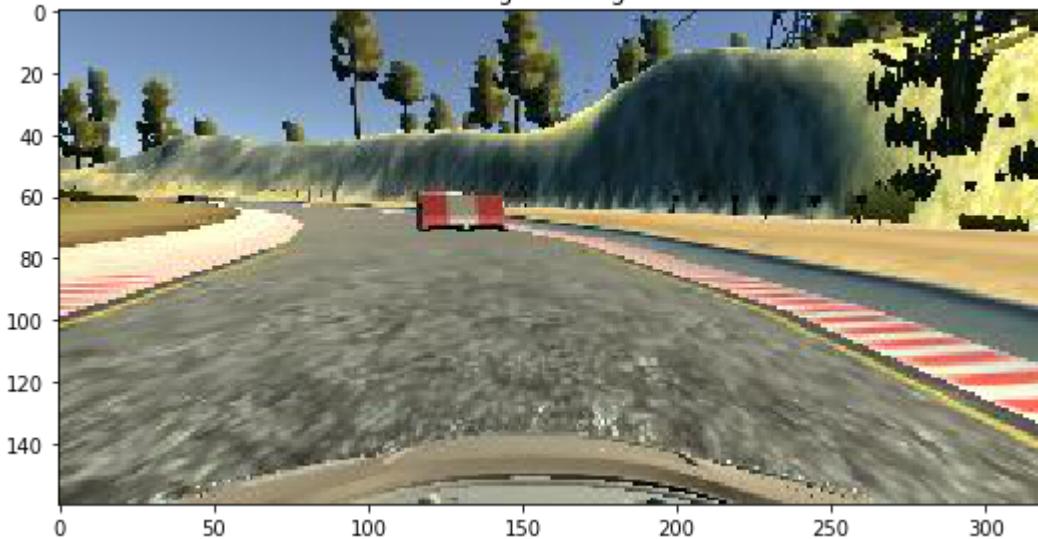
#brightnessimage
def img_random_brightness(image):
    brightness = iaa.Multiply((0.2, 1.2))
    image = brightness.augment_image(image)
    return image
image = image_paths[random.randint(0, 1000)]
original_image = mpimg.imread(image)
brightness_altered_image = img_random_brightness(original_image)
fig, axs = plt.subplots(1, 2, figsize=(15, 10))
fig.tight_layout()
axs[0].imshow(original_image)
axs[0].set_title('Original Image')
axs[1].imshow(brightness_altered_image)
axs[1].set_title('Brightness altered image ')

```

↪

Text(0.5, 1, 'Brightness altered image ')

Original Image

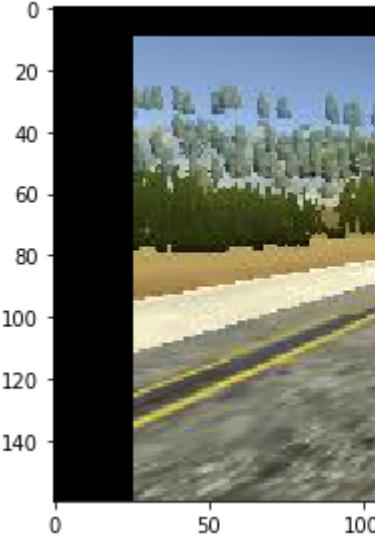
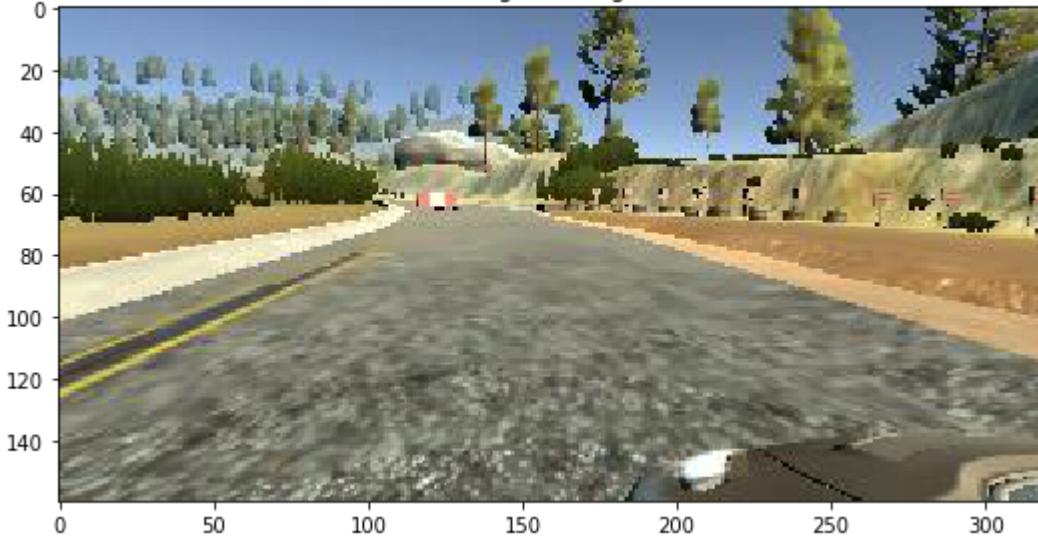


```
#panimage
```

```
def pan(image):
    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('Original Image')
axs[1].imshow(panned_image)
axs[1].set_title('Panned Image')
```

Text(0.5, 1, 'Panned Image')

Original Image



```
#flipimage
```

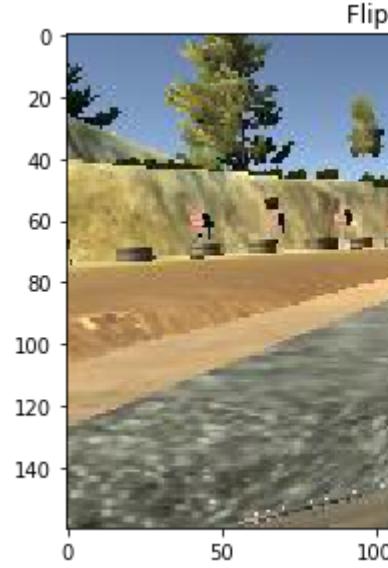
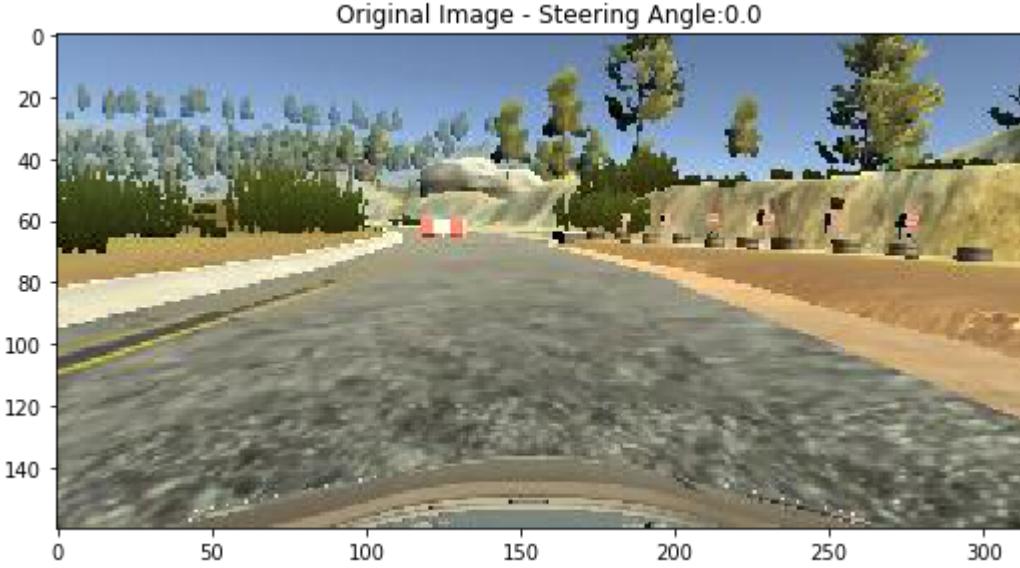
```
def img_random_flip(image, steering_angle):
    image = cv2.flip(image,1)
    steering_angle = -steering_angle
    return image, steering_angle
random_index = random.randint(0, 1000)
image = image_paths[random_index]
```

```

steering_angle = steerings[random_index]
original_image = mpimg.imread(image)
flipped_image, flipped_steering_angle = img_random_flip(original_image, steering_angle)
fig, axs = plt.subplots(1, 2, figsize=(15, 10))
fig.tight_layout()
axs[0].imshow(original_image)
axs[0].set_title('Original Image - ' + 'Steering Angle:' + str(steering_angle))
axs[1].imshow(flipped_image)
axs[1].set_title('Flipped Image - ' + 'Steering Angle:' + str(flipped_steering_angle))

```

↳ Text(0.5, 1, 'Flipped Image - Steering Angle:-0.0')



```
#augmentedimage
```

```

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 = 10
fig, axs = plt.subplots(nrow, ncol, figsize=(15, 50))
fig.tight_layout()
for i in range(10):
    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")

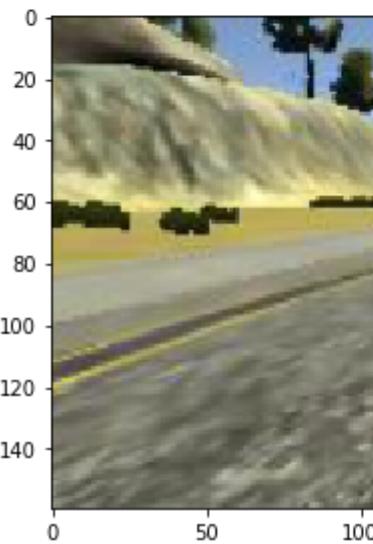
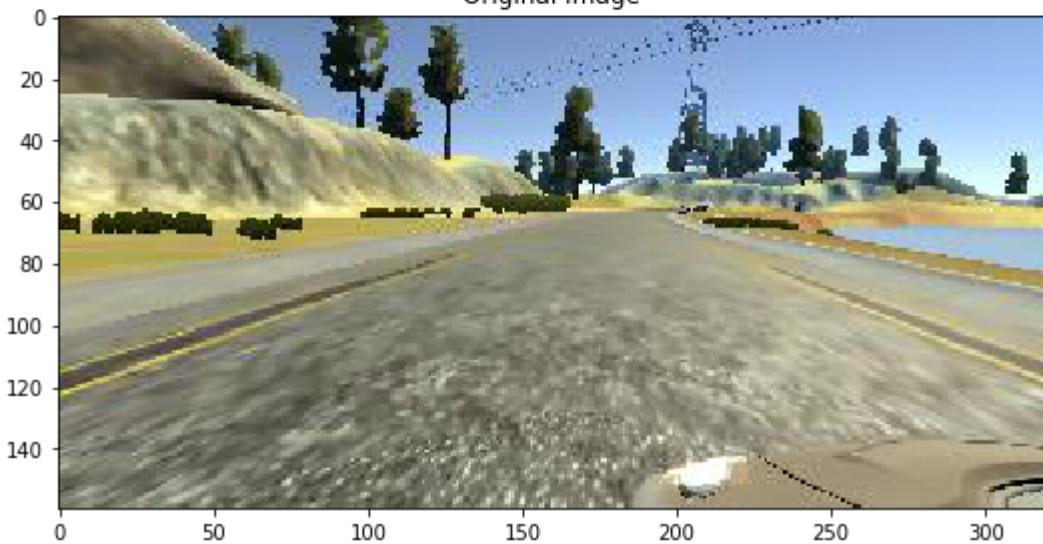
```

↳

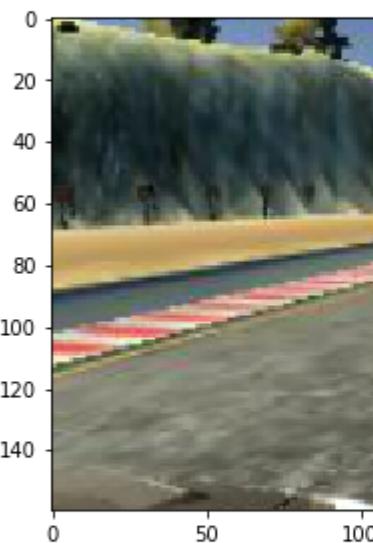
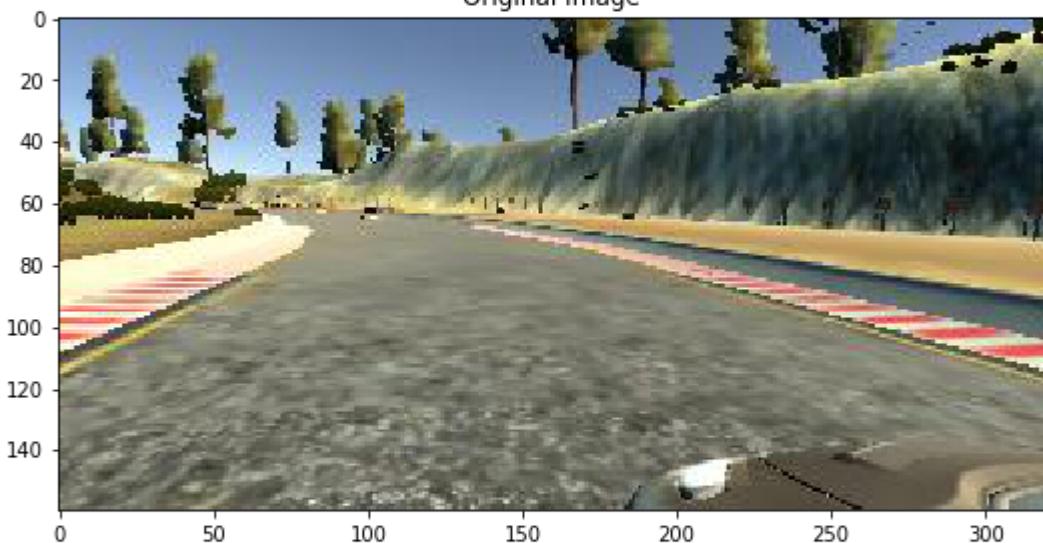
Original Image



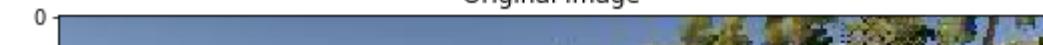
Original Image

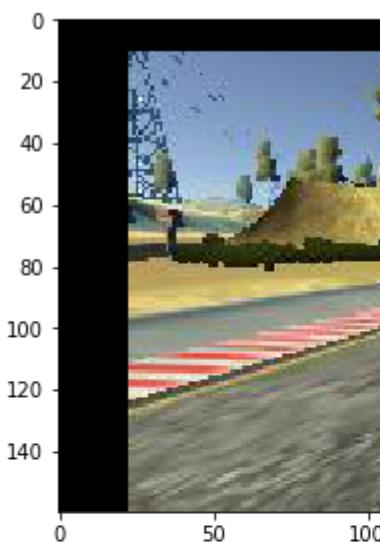
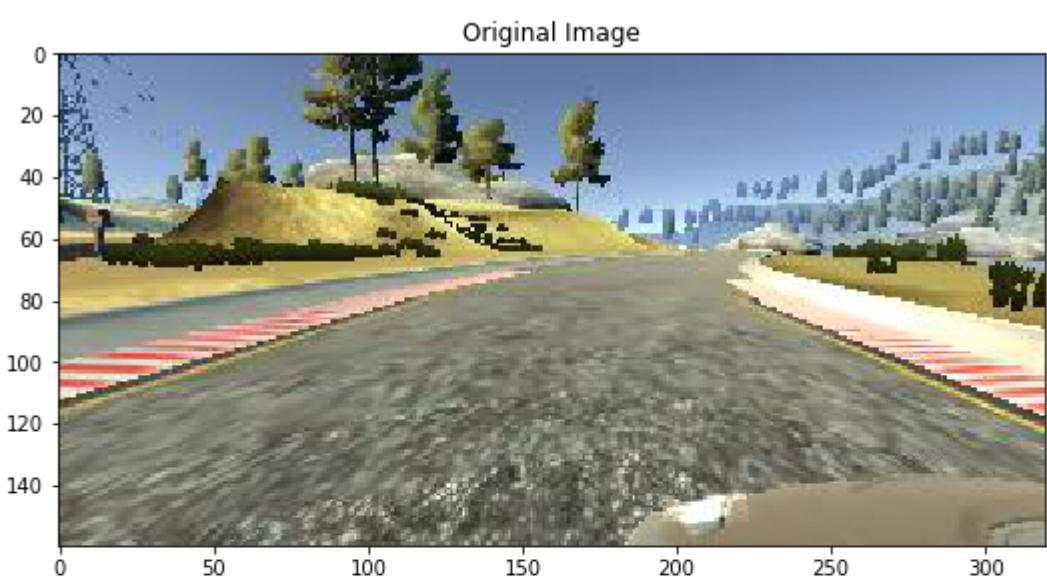
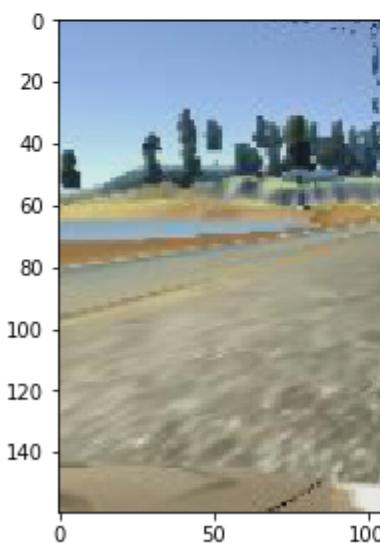
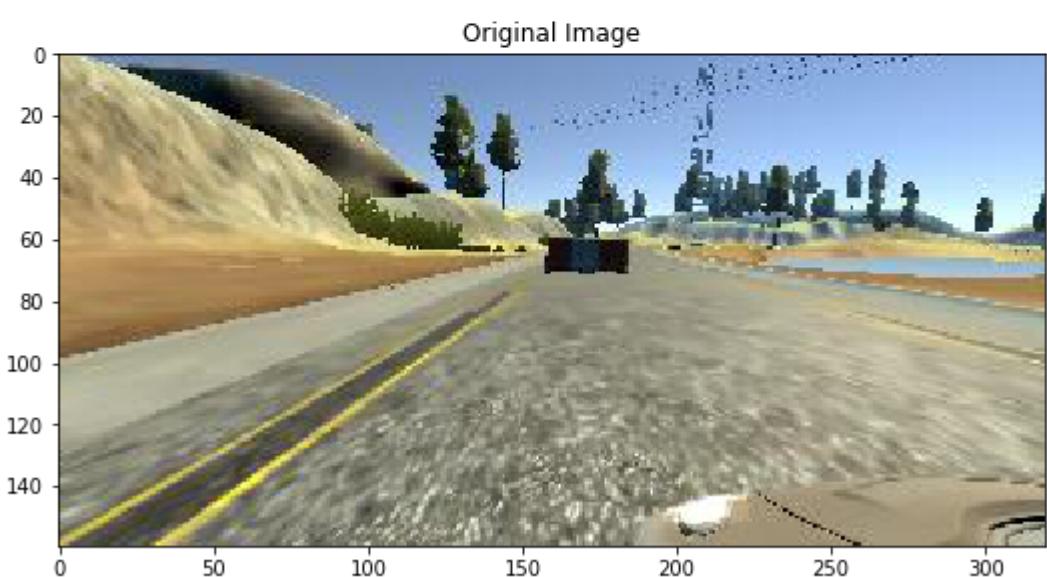
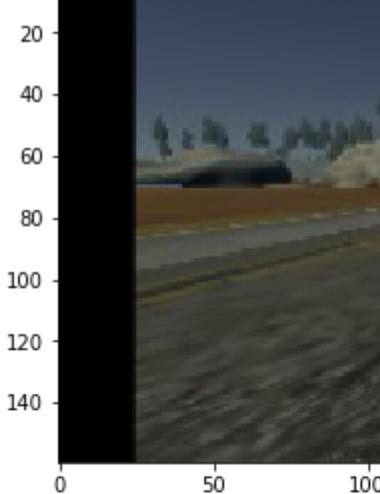
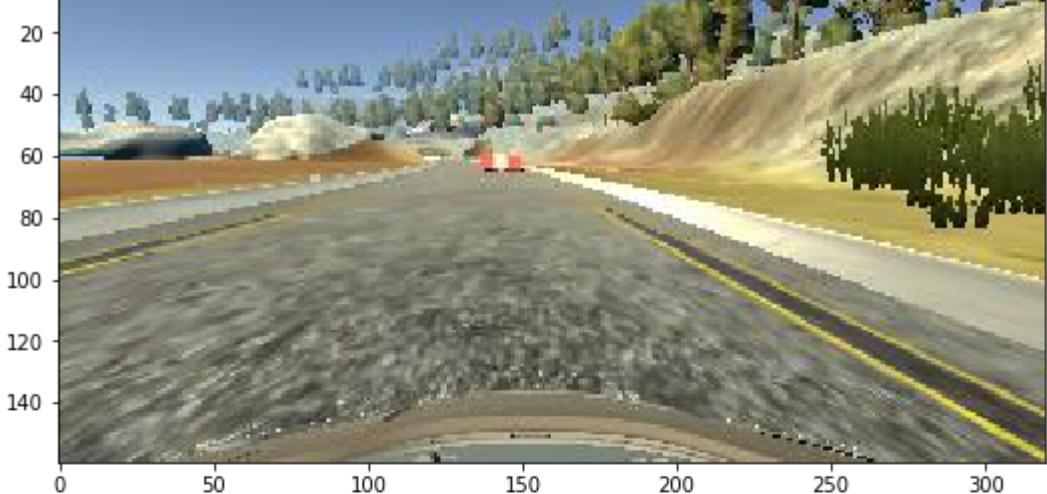


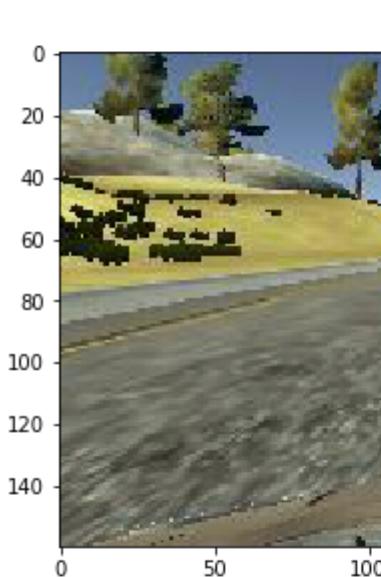
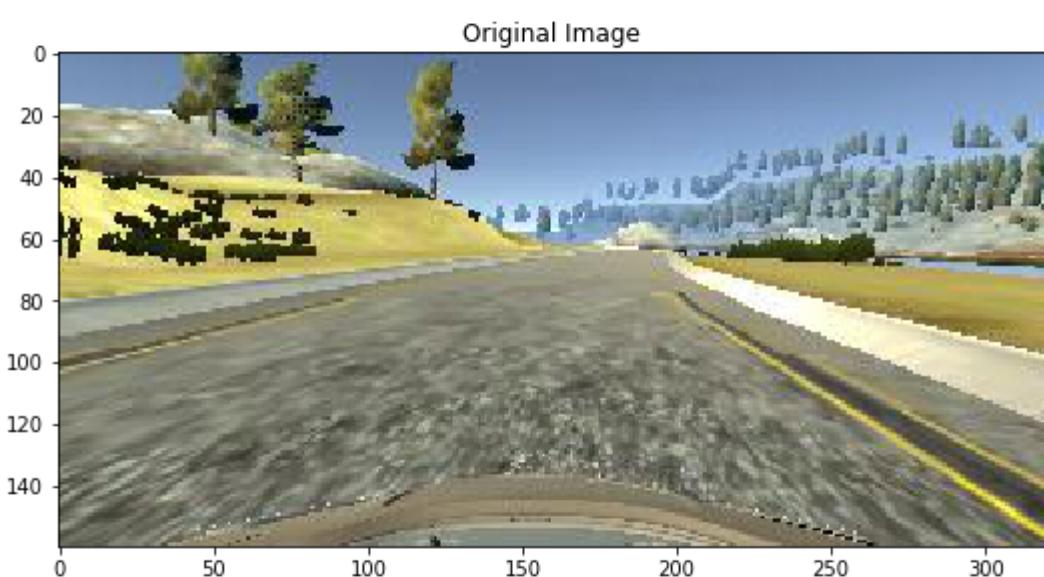
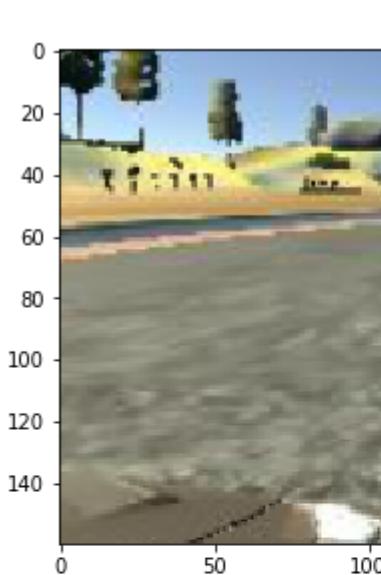
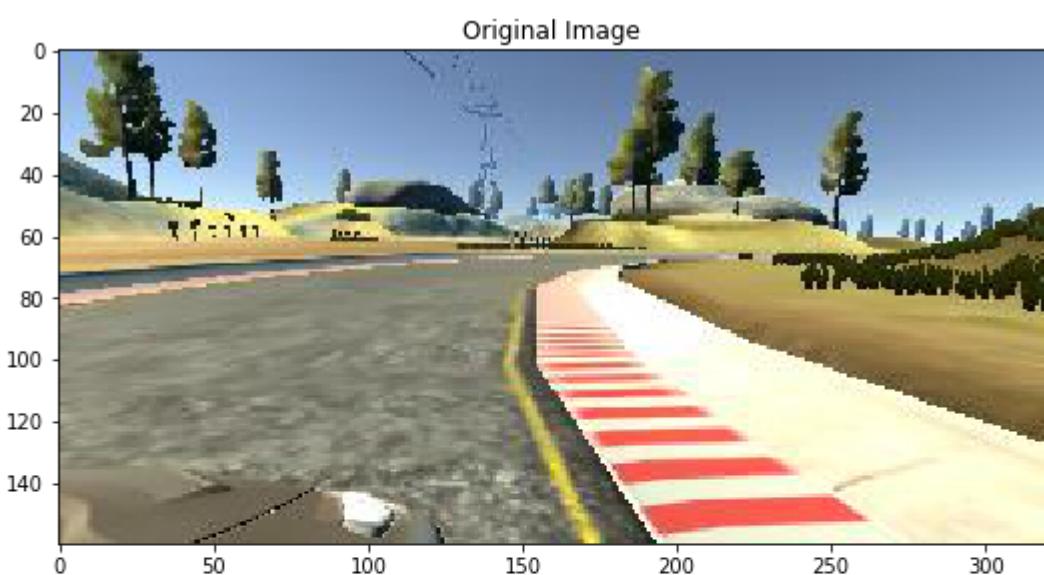
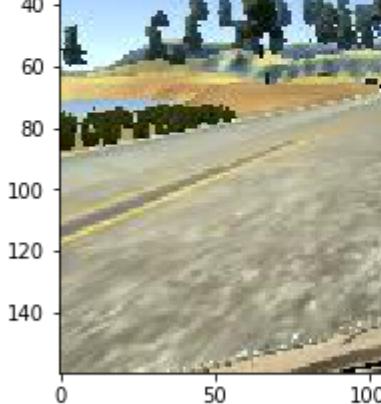
Original Image

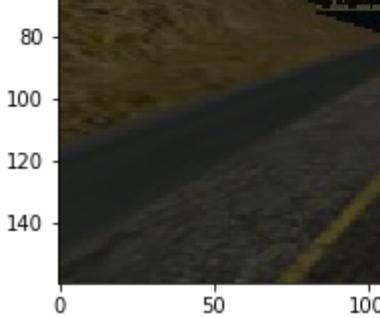


Original Image







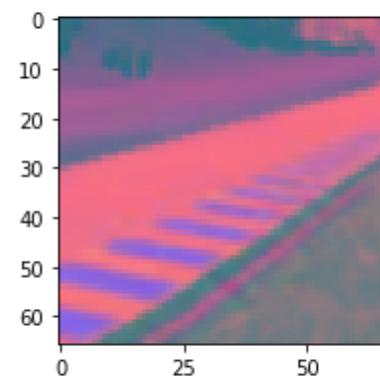


```
def img_preprocess (img):
    img = img[55:135,:,:]
    img = cv2.cvtColor(img, cv2.COLOR_RGB2YUV)
    img = cv2.GaussianBlur(img, (3, 3), 0)
    img = cv2.resize(img, (200, 66))
    img = img/255
    return img
```

```
image = image_paths[150]
original_image = mpimg.imread(image)
preprocessed_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('Original Image')
axs[1].imshow(preprocessed_image)
axs[1].set_title('processed Image')
```

↪ Text(0.5, 1, 'processed Image')

Original Image



```
#batchgenerator
def batch_generator(image_paths, steering_ang, batch_size, istraining):

    while True:
        batch_img = []
        batch_steering = []

        for i in range(batch_size):
            random_index = random.randint(0, len(image_paths) - 1)

            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]

    im = img_preprocess(im)
    batch_img.append(im)
    batch_steering.append(steering)
yield (np.asarray(batch_img), np.asarray(batch_steering))

```

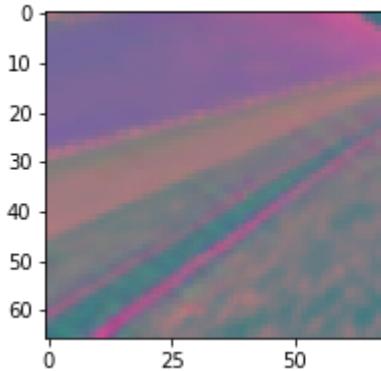
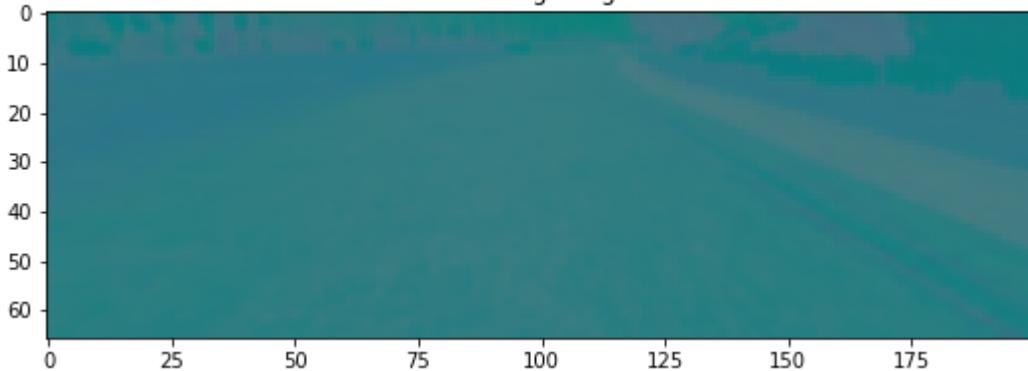
```

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, 'Validation Image')

Training Image



```

def nvidia_model():
    model = Sequential()
    model.add(Conv2D(26, 5, 5, subsample=(2, 2), input_shape=(66, 200, 3), activation='elu'))
    model.add(Conv2D(38, 5, 5, subsample=(2, 2), activation='elu'))
    model.add(Conv2D(52, 5, 5, subsample=(2, 2), activation='elu'))
    model.add(Conv2D(64, 3, 3, activation='elu'))
    model.add(Conv2D(128, 3, 3, activation='elu'))
    #model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(100, activation = 'elu'))
    #model.add(Dropout(0.5))
    model.add(Dense(50, activation = 'elu'))
    #model.add(Dropout(0.5))
    model.add(Dense(10, activation = 'elu'))
    #model.add(Dropout(0.5))
    model.add(Dense(1))

    optimizer = Adam(lr=1e-3)
    model.compile(loss='mse', optimizer=optimizer)
    return model

model = nvidia_model()
print(model.summary())

```

→

Model: "sequential_2"

Layer (type)	Output Shape	Param #
<hr/>		
conv2d_6 (Conv2D)	(None, 31, 98, 26)	1976
conv2d_7 (Conv2D)	(None, 14, 47, 38)	24738
conv2d_8 (Conv2D)	(None, 5, 22, 52)	49452
conv2d_9 (Conv2D)	(None, 3, 20, 64)	30016
conv2d_10 (Conv2D)	(None, 1, 18, 128)	73856
flatten_2 (Flatten)	(None, 2304)	0
dense_5 (Dense)	(None, 100)	230500
dense_6 (Dense)	(None, 50)	5050
dense_7 (Dense)	(None, 10)	510
dense_8 (Dense)	(None, 1)	11
<hr/>		

Total params: 416,109

Trainable params: 416,109

Non-trainable params: 0

None

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: UserWarning: Update your `Conv2D`  
    This is separate from the ipykernel package so we can avoid doing imports until  
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: UserWarning: Update your `Conv2D`  
    after removing the cwd from sys.path.  
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: UserWarning: Update your `Conv2D`  
    """  
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:6: UserWarning: Update your `Conv2D`  
    /usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:7: UserWarning: Update your `Conv2D`  
        import sys
```

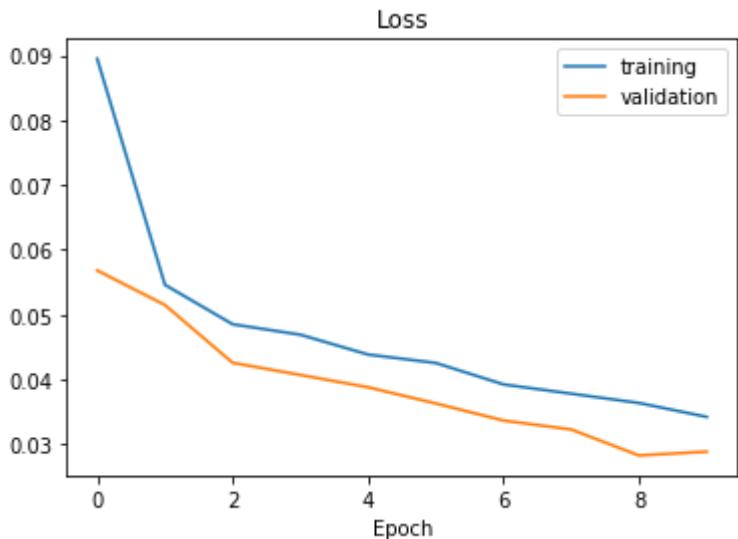
```
history = model.fit_generator(batch_generator(X_train, y_train, 100, 1),  
                             steps_per_epoch=300,  
                             epochs=10,  
                             validation_data=batch_generator(X_valid, y_valid, 100, 0),  
                             validation_steps=200,  
                             verbose=1,  
                             shuffle = 1)
```



```
Epoch 1/10
300/300 [=====] - 485s 2s/step - loss: 0.0895 - val_loss: 0.0568
Epoch 2/10
300/300 [=====] - 474s 2s/step - loss: 0.0546 - val_loss: 0.0515
Epoch 3/10
300/300 [=====] - 472s 2s/step - loss: 0.0485 - val_loss: 0.0426
Epoch 4/10
300/300 [=====] - 469s 2s/step - loss: 0.0469 - val_loss: 0.0407
Epoch 5/10
300/300 [=====] - 471s 2s/step - loss: 0.0439 - val_loss: 0.0388
Epoch 6/10
300/300 [=====] - 465s 2s/step - loss: 0.0426 - val_loss: 0.0363
Epoch 7/10
300/300 [=====] - 470s 2s/step - loss: 0.0393 - val_loss: 0.0337
Epoch 8/10
300/300 [=====] - 469s 2s/step - loss: 0.0378 - val_loss: 0.0323
Epoch 9/10
300/300 [=====] - 468s 2s/step - loss: 0.0364 - val_loss: 0.0283
Epoch 10/10
300/300 [=====] - 464s 2s/step - loss: 0.0342 - val_loss: 0.0289
```

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['training', 'validation'])
plt.title('Loss')
plt.xlabel('Epoch')
```

↪ Text(0.5, 0, 'Epoch')



```
model.save('modifiedarchjan20.h5')
```

```
#from google.colab import files
#files.download('model.h5')
```

