Natural Disasters Intensity Analysis And Classification Using Artificial Intelligence

Image Preprocessing

Team ID : PNT2022TMID52367

Introduction:

Pre-processing is a common name for operations with images at the lowest level of abstraction — both input and output are intensity images. These iconic images are of the same kind as the original data captured by the sensor, with an intensity image usually represented by a matrix of image function values (brightnesses).

The aim of pre-processing is an improvement of the image data that suppresses unwilling distortions or enhances some image features important for further processing, although geometric transformations of images (e.g. rotation, scaling, translation) are classified among pre-processing methods here since similar techniques are used.

# Importing the Dataset

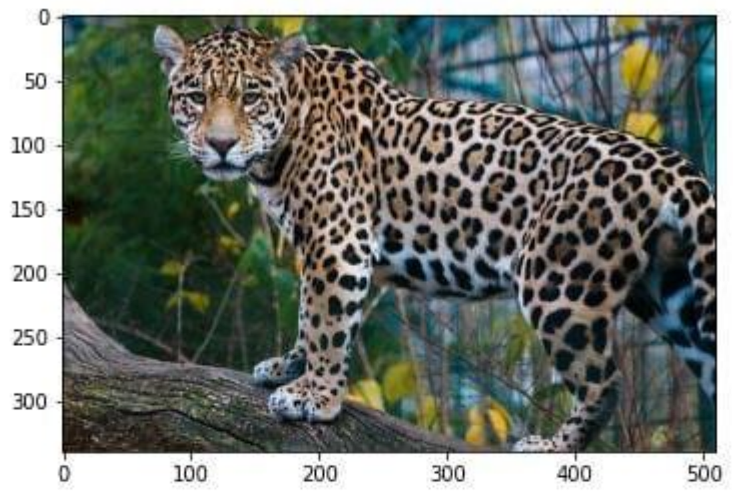
```
# importing libraries
import tensorflow
import keras
import os
import glob
from skimage import io
import random
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inlin
```

```
# Importing and Loading the data into a data frame

dataset_path = '/content/drive/MyDrive/Animals'
class_names = ['Cheetah', 'Jaguar', 'Leopard', 'Lion','Tiger']

# apply glob module to retrieve files/pathnames

animal_path = os.path.join(dataset_path, class_names[1], '*')
animal_path = glob.glob(animal_path)
```

To verify if our data is properly loaded, we shall try accessing an image file from the dataset:

```
# accessing an image file from the dataset classes
image = io.imread(animal_path[4])

# plotting the original image
i, (im1) = plt.subplots(1)
i.set_figwidth(15)
im1.imshow(image)
```

By executing the above code block, we shall randomly print an image from the dataset.

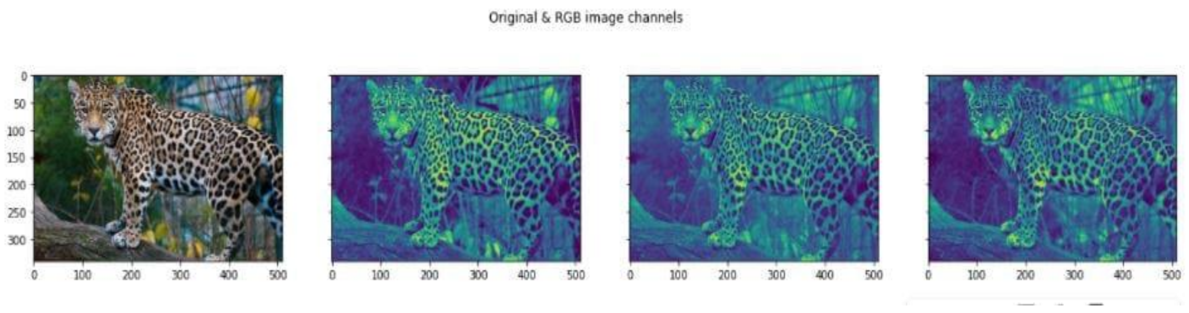


# Configure ImageDataGenerator

let’s first explore the RGB channels of our original image;

```
# plotting the original image and the RGB channels

i, (im1, im2, im3, im4) = plt.subplots(1, 4, sharey=True)
i.set_figwidth(20)

im1.imshow(image)  #Original image
im2.imshow(image[:, : , 0]) #Red
im3.imshow(image[:, : , 1]) #Green
im4.imshow(image[:, : , 2]) #Blue
i.suptitle('Original & RGB image channels')
```

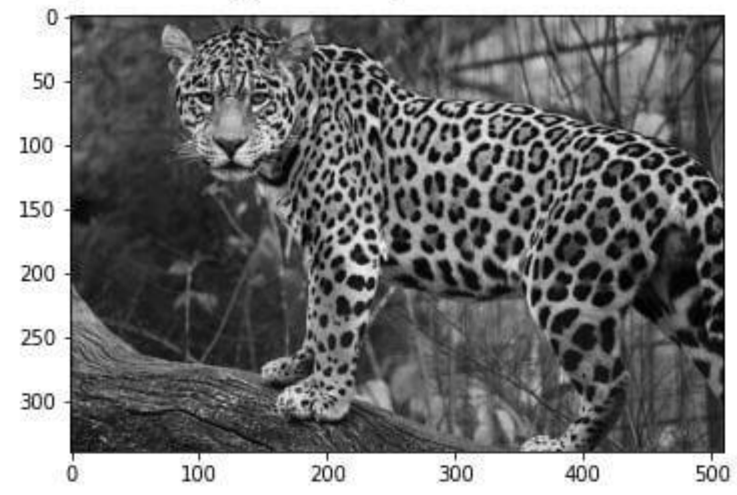
output



Execute the code below to convert the original image to grayscale:

```
gray_image = skimage.color.rgb2gray(image)
plt.imshow(gray_image, cmap = 'gray')
```

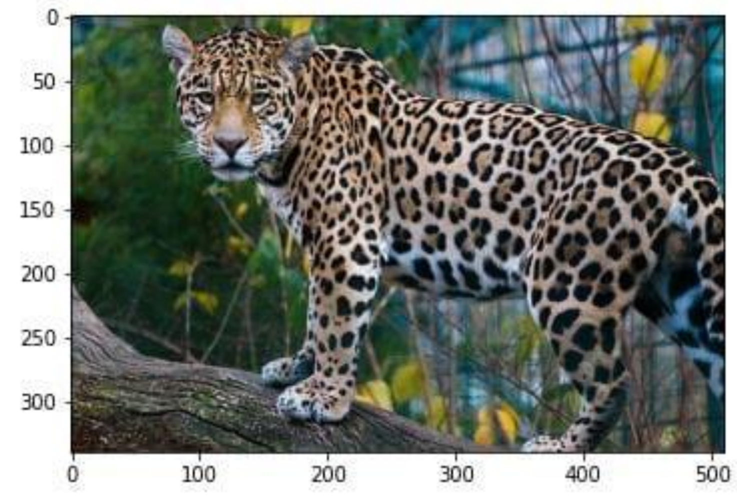
```
<matplotlib.image.AxesImage at 0x7f9159f08790>
```



Let’s write the code below to normalize our data.

```
norm_image = (gray_image - np.min(gray_image)) /
(np.max(gray_image) - np.min(gray_image))
plt.imshow(norm_image)
```

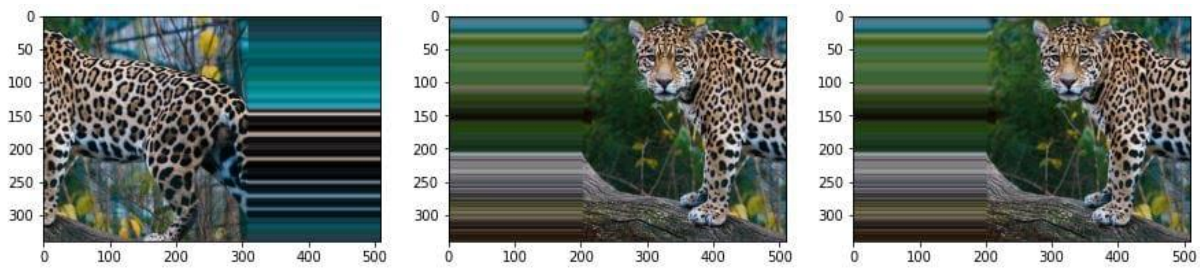
```
<matplotlib.image.AxesImage at 0x7f9159ef38d0>
```



let’s look at an example below that shifts horizontally:

```
# import libraries

from numpy import expand_dims
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator

# convert to numpy array
data = img_to_array(image)
```

```
# expand dimension to one sample
samples = expand_dims(image, 0)

# create image data augmentation generator
datagen = ImageDataGenerator(width_shift_range=[-200,200])

# create an iterator
it = datagen.flow(samples, batch_size=1)
fig, im = plt.subplots(nrows=1, ncols=3, figsize=(15,15))

# generate batch of images
for i in range(3):

    # convert to unsigned integers
    image = next(it)[0].astype('uint8')

    # plot image
    im[i].imshow(image)
```



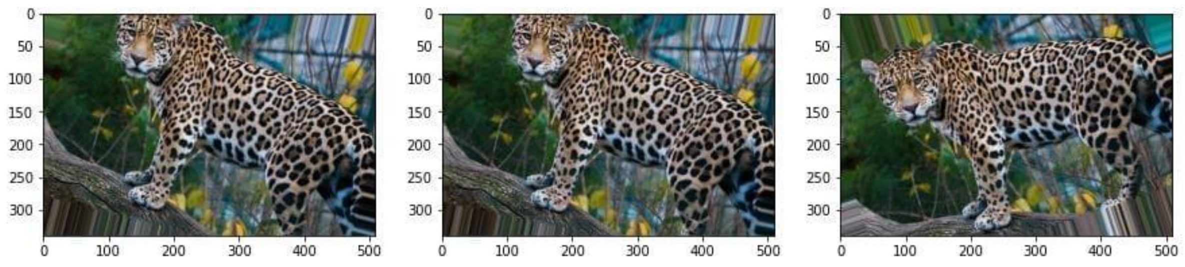
# Applying Image Data

This process involves rotating an image by a specified degree.

From the code above, change the **ImageDataGenerator** parameters, as shown below:

```
datagen = ImageDataGenerator(rotation_range=20,
fill_mode='nearest')
```

Output

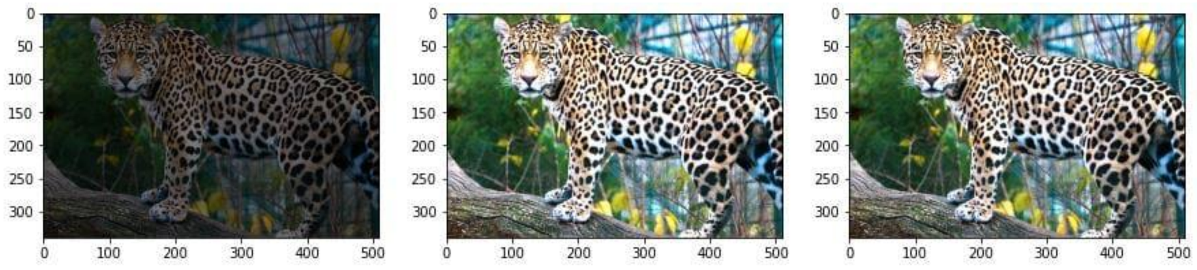


This is the process of increasing or decreasing image contrast.

From the code above, change the **ImageDataGenerator** parameters, as shown below:

```
datagen = ImageDataGenerator(brightness_range=[0.5,2.0])
```

Output



conclusion:

Having explored the popular and commonly used image preprocessing techniques, what now remains is modeling your machine learning models to the desired level of high accuracy and performance. Thus, we are now ready to jump into building custom computer vision projects.