Image Processing with Keras in Python

rn540

In this article, we are doing Image Processing with Keras in Python. Keras API is a deep learning library that provides methods to load, prepare and process images.

We will cover the following points in this article:

- Load an image
- Process an image
- Convert Image into an array and vice-versa
- Change the color of the image
- Process image dataset

Load the Image

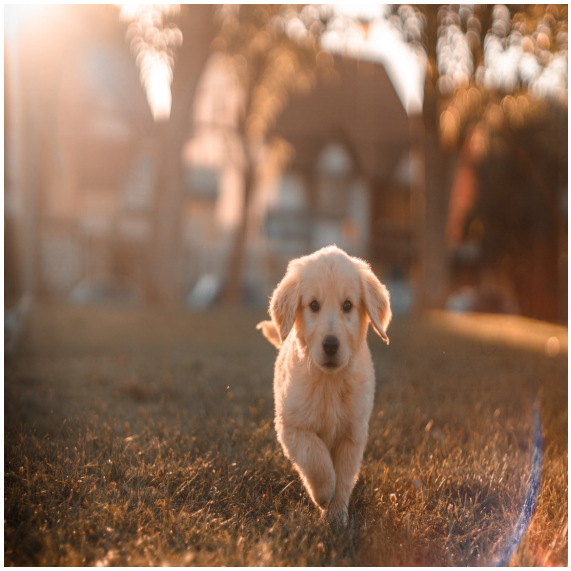
In Keras, load_img() function is used to load image. The image loaded using load_img() method is PIL object. Certain information can be accessed from loaded images like image type which is PIL object, the format is JPEG, size is (6000,4000), mode is RGB, etc. We are using dog images throughout the article.

Python3


```
import keras
from keras.preprocessing.image import load_img

# load the image
img = load_img('dog.jpg')

# find more about the image
print(type(img))
print(img.format)
print(img.mode)
print(img.size)

# show the image
img.show()
```

Output:

```
<class 'PIL.JpegImagePlugin.JpegImageFile'>
JPEG
RGB
(6000, 4000)
```



Resize an Image

We can perform certain functions on the image like resizing it, changing its color, convert into an array, etc before training any model. To resize the shape of the image, resize() method is invoked on the image. The size in which we want to convert the image should be iterable.

Python3

```
img = load_img('dog.jpg')

# change image size
image = img.resize([30, 30])

# print new image size
print(image.size)
```

Output:

```
(30, 30)
```

We can see in the output, 30 x 30 is the new size of the image.

Convert image into an array

There is img_to_array() method to convert images into array, and array_to_img() method to convert image array back to image. In the below example, we are just accessing the 0^{th} index of the image array. We can get information like image array shape, type, etc. When the image array is converted to an image, it again is a PIL object.

Python3

```
# convert image into array
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import array_to_img

# convert to numpy array
img_array = img_to_array(img)
print(img_array[0])
print(img_array.dtype)
print(img_array.shape)

# convert back to image
img = array_to_img(img_array)
print(type(img))
```

Output:

```
[[244. 244. 244.]
 [244. 244. 244.]
 [244. 244. 244.]
 ...
 [117.  74.  42.]
 [115.  71.  42.]
 [112.  68.  39.]]
float32
(4000, 6000, 3)
<class 'PIL.JpegImagePlugin.JpegImageFile'>
```

Change the color of the Image

To convert the colorful images into grayscale, just set grayscale = True in load_img() method. To save the converted images we need to convert the image into an array, otherwise the save_img() method will throw an error. After converting the image into grayscale image we see it still shows mode RGB, and the size is the same.

Python3

```
from keras.preprocessing.image import save_img
from keras.preprocessing.image import img_to_array

# load image as grayscale
img = load_img('dog.jpg', grayscale=True)

# convert image to a numpy array
img_array = img_to_array(img)

# save the image with a new filename
save_img('dog_grayscale.jpg', img_array)

# load the image to confirm it was saved
# correctly
img = load_img('dog_grayscale.jpg')
print(type(img))
print(img.format)
print(img.mode)
print(img.size)
img.show()
```

Output:

```
<class 'PIL.Image.Image'>
None
RGB
(6000, 4000)
```

Process an Image dataset

To load the images from the image dataset, the simple method is to use load_data() on the image dataset. We are using mnist dataset which is already available in Keras. It will give in return x_train, y_train, x_test, and y_test. The x_train and y_train will be used to train the model and x_test and y_test will be used for testing purposes. We can reshape all the images inside the dataset using reshape() method, and define what type of images should be like 'float64' or 'float32'.

Python3

```
from keras.datasets import mnist

(X_train, Y_train), (X_test, Y_test) = mnist.load_data()

# reshape the image
images = X_train.reshape(-1, 28, 28, 1).astype('float64')

print(images.shape)
print(type(images))
print(images.size)
```

Output:

```
(60000, 28, 28, 1)
<class 'numpy.ndarray'>
47040000
```

We see in the above output, 60000 images in mnist have been reshaped into 28 x 28 size and images are of type numpy n-dimensional array.

Last Updated : 12 Oct, 2022

2

Similar Reads

1. Python Keras | keras.utils.to_categorical()
2. keras.fit() and keras.fit_generator()
3. Point Processing in Image Processing using Python-OpenCV
4. Image Processing in Java - Colored Image to Grayscale Image Conversion
5. Image Processing in Java - Colored image to Negative Image Conversion
6. Image Processing in Java - Colored Image to Sepia Image Conversion
7. Python | Image Classification using Keras
8. How to Normalize, Center, and Standardize Image Pixels in Keras?
9. Image processing with Scikit-image in Python
10. Image Processing in Java - Colored to Red Green Blue Image Conversion

Related Tutorials