processingimagedatafordl

April 9, 2024

Things I done here: - Reading an image file and converting it to a numpy array - Resizing an image - RGB to Grayscale conversion

[2]:
```
# getting an image using web get

!wget 'https://e1.pxfuel.com/desktop-wallpaper/978/447/
↪desktop-wallpaper-puppies-cellphone-cute-puppies-phone-thumbnail.jpg'
```

```
--2024-04-09 08:46:52--  https://e1.pxfuel.com/desktop-
wallpaper/978/447/desktop-wallpaper-puppies-cellphone-cute-puppies-phone-
thumbnail.jpg
Resolving e1.pxfuel.com (e1.pxfuel.com)… 104.21.12.22, 172.67.151.78,
2606:4700:3037::ac43:974e, …
Connecting to e1.pxfuel.com (e1.pxfuel.com)|104.21.12.22|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 35655 (35K) [image/jpeg]
Saving to: ‘desktop-wallpaper-puppies-cellphone-cute-puppies-phone-
thumbnail.jpg’

desktop-wallpaper-p 100%[===================>]  34.82K  --.-KB/s    in 0.001s

2024-04-09 08:46:53 (56.9 MB/s) - ‘desktop-wallpaper-puppies-cellphone-cute-
puppies-phone-thumbnail.jpg’ saved [35655/35655]
```

Libraries that can be used for image processing

1. matplotlib.image
2. Pillow
3. OpenCV(cv2)

[10]:
```
# importing the image module from matplotlib library
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
```

[11]:
```
# loading an image module through matplotlib.image module
img = mpimg.imread('/content/dog_image.jpg')
```

[12]:
```
type(img)
```

[12]:
```
numpy.ndarray
```

[13]:
```
# shape of image in form of array
img.shape
```

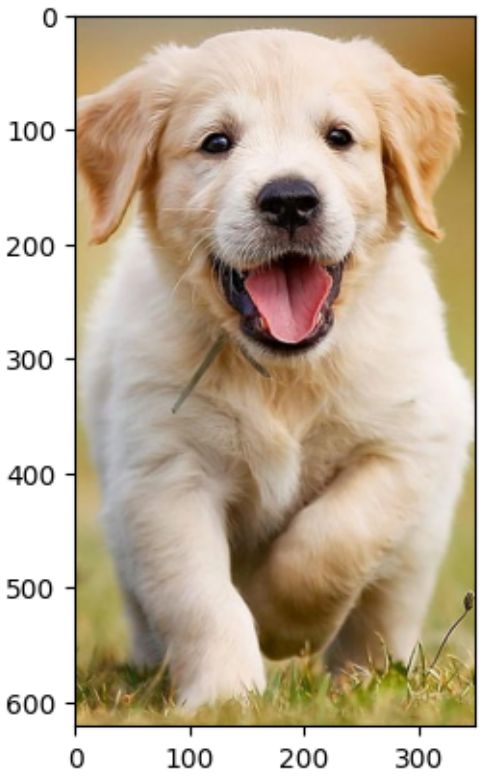
[13]:
```
(622, 350, 3)
```

[14]:
```
# print the numpy array the image contain
print(img)
```

```
[[[150 146 121]
  [150 146 121]
  [150 146 121]
  …
  [149 111  64]
  [149 111  64]
  [148 110  63]]

 [[151 145 121]
  [151 145 121]
  [151 145 121]
  …
  [151 111  62]
  [149 111  62]
  [148 110  61]]

 [[153 145 122]
  [153 145 122]
  [153 145 122]
  …
  [151 111  62]
  [148 110  61]
  [148 110  61]]


 …

 [[206 176 104]
  [204 174 102]
  [203 173 103]
  …
  [181 157  83]
  [178 156  81]
  [178 156  81]]

 [[203 173 101]
```

```
 [201 171  99]
 [199 169  99]
 …
 [181 157  83]
 [179 157  82]
 [178 156  81]]

[[197 167  95]
 [196 166  94]
 [194 164  94]
 …
 [184 160  86]
 [182 160  87]
 [181 159  86]]]
```

[15]:
```
# displaying the image from numpy array

img_plot = plt.imshow(img)
plt.show()
```



Resizing the Image using Pillow Library

[16]:
```
from PIL import Image
```

[42]:
```
img = Image.open('/content/dog_image.jpg')
img_resized = img.resize((200,200))
```

[27]:
```
img_resized.save('dog_image_resized.jpg')
```

[28]:
```
# displaying the image from numpy array
img_res = mpimg.imread('/content/dog_image_resized.jpg')
img_res_plot = plt.imshow(img_res)
plt.show()
```

[29]:
```
#now checking the shape of new image
img_res.shape
```

[29]:
```
(200, 150, 3)
```

Converting RGB image to Grayscale image using OpenCV

[30]:
```
# importing the OpenCV library
import cv2
```

[31]:
```
img = cv2.imread('/content/dog_image.jpg')
```

[32]:
```
type(img)
```

[32]:
```
numpy.ndarray
```

[33]:
```
img.shape
```

[33]:
```
(622, 350, 3)
```

[36]:
```
grayscale_image = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
```

[37]:
```
type(grayscale_image)
```

[37]:
```
numpy.ndarray
```

[38]:
```
grayscale_image.shape
```

[38]:
```
(622, 350)
```

Thus, we saw this image has color channel is only one not three like RGB image - It's been converted just due to make process easier as it only contain one channel.

- cv2.imshow() will display the image. But, this will not be allowed in Google Colab.

```
from google.colab.patches import cv2_imshow
```

[39]:
```
from google.colab.patches import cv2_imshow
```

[41]:
```
#displaying the image
cv2_imshow(grayscale_image)
```

[43]:
```
# saving the grayscale image
cv2.imwrite( 'dog _ grayscale _ image .jpg', grayscale_image)
```

[43]: True

This is how we process image in Deep Learning Projects.

Hasrat Ali
Thank You:)