Experiment No.10
Program to demonstrate use of numpy array for working with images
Date of Performance:
Date of Submission:

Experiment No 10:

Aim: Program to demonstrate use of numpy array for working with

images

Theory

Images are an easier way to represent the working model. In Machine Learning, Python uses the image data in the format of Height, Width, Channel format. i.e. Images are converted into Numpy Array in Height, Width, Channel format.

Modules Needed:

NumPy: By default in higher versions of Python like 3.x onwards, NumPy is available and if not available(in lower versions), one can install by using

pip install numpy

Pillow: This has to be explicitly installed in later versions too. It is a preferred image manipulation tool. In Python 3, Pillow python library which is nothing but the upgradation of PIL only. It can be installed using

pip install Pillow

One can easily check the version of installed Pillow by using the

below code **import** PIL

```
print('Installed Pillow Version:', PIL.__version__)
```

Output:

Installed Pillow Version: 7.2.0

Loading the images via Pillow Library

Let us check for an image that is in the PNG or JPEG format. The image can be referred via its path. Image class is the heart of PIL. It has open() function which opens up an image and digital file format can be retrieved as well as pixel format.

Converting an image into NumPy Array

Python provides many modules and API's for converting an image into a NumPy

array. Let's discuss a few of them in detail.

Using NumPy module

Numpy module in itself provides various methods to do the same. These methods are – **Method 1: Using asarray() function**

asarray() function is used to convert PIL images into NumPy arrays. This function converts the input to an array

Method 2: Using numpy.array() function

By using numpy.array() function which takes an image as the argument and converts to NumPy array

In order to get the value of each pixel of the NumPy array image, we need to print the retrieved data that got either from asarray() function or array() function.

Getting back the image from converted Numpy Array

Image.fromarray() function helps to get back the image from converted numpy array. We get back the pixels also same after converting back and forth. Hence, this is very much efficient

PROGRAM:

```
from PIL import Image

from numpy import as array

# load the image

image = Image.open('demopic.jpg')

# convert image to numpy array

data = asarray(image)

print(type(data))
```

```
# summarize shape
print(data.shape)
print(data)
image2 = Image.fromarray(data)
print(type(image2))
# summarize image details
print(image2.mode)
print(image2.size)
```

OUTPUT:

```
=RESTART:
C:\Users\admin\AppData\Local\Programs\Python\Python310\img2.py
==========================================

<class 'numpy.ndarray'>
(377, 271, 3)
[[[169 180 114]
 [167 177 114]
 [167 173 113]
 ...
 [152 133 93]
```

[151 132 92]

[150 130 93]]

[[161 171 110]

[160 170 110]

[158 166 109]

...

[151 132 92]

[149 130 90]

[148 128 91]]

[[150 159 106]

[148 156 105]

[147 153 105]

...

[152 133 93]

[150 131 91]

[149 131 93]]

...

[[184 124 72]

[182 125 72]

[180 122 72]

...

[241 221 13]

[240 222 16]

[241 223 19]]

[[181 121 69]

[179 122 69]

[177 119 69]

...

[229 211 7]

[225 208 4]

[222 203 3]]

[[182 122 70]

[181 124 71]

[180 122 72]

...

[222 205 4]

[222 206 5]

[219 201 5]]]

<class
'PIL.Image.Image'
> JPEG

RGB

(271, 377)

Conclusion: , the experiment effectively showcased the practical application of numpy arrays in image manipulation, underscoring their efficiency and versatility in processing visual data. By implementing fundamental numpy functions, the experiment demonstrated how these arrays can be harnessed to perform various image operations with ease and precision, providing a solid foundation for further exploration and development in the field of image processing.