HW 00 – REPORT

소속 : 정보컴퓨터공학부

학번 : 2021924624

이름 : Nemekhbayar Nomin

1.Introduction

Practice objectives and theoretical background skills (page 1-2)

Pillow as PIL is a fork of an older library called PIL which is Python Imaging Library as they say. It is known as the main library for opening, manipulating and saving image file formats of python programming language.

Numpy as np is a library for the Python programming language to support large, multi-dimensional arrays and matrices. What makes different Numpy array from Python array and lists is that, Numpy arrays are faster, eats less memory to store data and great for numerical computations.

With Image from PIL library:

- Image.open() can be used to open an image file with directory path.
- Image.convert returns a converted copy of the image and converts an image from one mode to another mode.
- image.show() is a method for displaying the image.
- image.crop() is a method for cropping an image with the specific coordinates.
- image.save() takes arguments of a name which the image to be saved with and file type.
- Image.fromarray() takes an array and converts it into image type.

Above examples of some functions are the basic methods that were used in the beginning of the code.

Let’s go through the methods of NumPy as np:

- np.asarray() is used for taking an array format of the image in this case.
- np.mean() is used for computing the mean value of the array.
- np.arange() generates a sequence of numbers within the specified range which, in this case, between 0 and 256
- np.tile(array, repetitions) constructs a new array by repeating array per

repetitions.

2. Subject

Practice and results description (more than 2 pages)

I have runned the code on Jupyter Notebook without any issues nor errors.

Aside from PIL and NumPy libraries, mathematical functions were used in the code for computational use such as min() and np.ndarray.astype() casts the copy of the array to a specified type.

We takes an image with (750, 599) pixels size and crops the head section with (150,150) square sized to an image named chipmunk_head with png image type.

Then, converting the head image to an array, in the loops, image has become brighter by taking minimum value of each pixels after adding 50 values to pixels value or 255 (whitest value). It is now the chipmunk_head_bright.png image after it is converted to an image type using Image.fromarray(‘arr’) .

With the copy of the previous array, it is multiplied by 0.5 which results in darker pixels with 0 being dark and 255 being the white. Also, ‘im4_array.astype('uint8')’ is

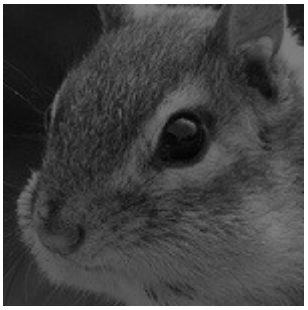
necessary to convert the any fractional values to integer type, otherwise an error raises due to Image.fromarray nature of taking only integer types as input.

```
: # im4_array = im4_array.astype('uint8')
  # im4_array
```

```
: im4 = Image.fromarray(im4_array)
  im4.save('chipmunk_head_dark.png','PNG')
  im4.show()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~/anaconda3/lib/python3.10/site-packages/PIL/PngImagePlugin.py:1299, in _save(im, fp,
filename, chunk, save_all)
   1298 try:
-> 1299     rawmode, mode = _OUTMODES[mode]
   1300 except KeyError as e:

KeyError: 'F'

The above exception was the direct cause of the following exception:

OSError                                   Traceback (most recent call last)
Cell In[18], line 2
      1 im4 = Image.fromarray(im4_array)
----> 2 im4.save('chipmunk_head_dark.png','PNG')
      3 im4.show()

File ~/anaconda3/lib/python3.10/site-packages/PIL/Image.py:2431, in Image.save(self, fp, f
ormat, **params)
   2428         fp = builtins.open(filename, "w+b")
   2430 try:
-> 2431     save_handler(self, fp, filename)
   2432 except Exception:
   2433     if open_fp:

File ~/anaconda3/lib/python3.10/site-packages/PIL/PngImagePlugin.py:1302, in _save(im, fp,
filename, chunk, save_all)
   1300 except KeyError as e:
   1301     msg = f"cannot write mode {mode} as PNG"
-> 1302     raise OSError(msg) from e
   1304 #
   1305 # write minimal PNG file
   1307 fp.write(_MAGIC)

OSError: cannot write mode F as PNG
```

With ‘im4_array.astype('uint8')’:



Gradient image:

In the last lines of the given code, it is seen that we can generate gradient image with simple array methods.

grad = np.arange(0,256) will be just simple 1D array with values from 0 to 255.

np.tile() now takes the grad array and replicates it 256 times along the first axis

which is row, and 1 times along the second axis, column.

```
24]: grad = np.arange(0,256)
     grad = np.tile(grad,[256,1])
     grad
```

```
24]: array([[  0,   1,   2, ..., 253, 254, 255],
            [  0,   1,   2, ..., 253, 254, 255],
            [  0,   1,   2, ..., 253, 254, 255],
            ...,
            [  0,   1,   2, ..., 253, 254, 255],
            [  0,   1,   2, ..., 253, 254, 255],
            [  0,   1,   2, ..., 253, 254, 255]])
```

Shape of the resulting grad array is (256,256).

```
: grad.shape
```

```
: (256, 256)
```

3. Conclusion

Discussion and Conclusion (page 1)

The conclusion that can be drawn out of the exercise code, is that we can manipulate images using NumPy array, some simple mathematical computations and PIL libraries.

In addition to that, input types should be carefully integrated, in order to match the function syntaxes and to not raise Traceback error or exceptions. Parameters also, should be studied from the library docs or websites and for future purposes, it should be remembered for faster typing. By practicing, I think functions become familiar and likely to be used easily.

Lastly, I believe I got more familiar with pixel-level of image and its array representation more than before, by the end of this assignment. It was simple and easy, yet it gave the most approachable way to get to know about pixel values and what are the numbers being recognized by the computers behind the image we see by human eyes.