Image and Video Processing Lab
Lab 1: Familiarization to Image Processing using Python

Aruna Shaju Kollannur (SC21M111)

Sub: Image and Video Processing Lab
Date of Lab sheet: August 31, 2021
Department: Avionics(DSP)
Date of Submission: September 6, 2021

Question 1(a): Find the largest and smallest element in a matrix and print the values along with their position.

Aim

To find the largest and smallest elements along with the position from a matrix

Discussion

A nested loop is run and elements in the inputted matrix is compared with each other to find the smallest and largest elements along with its position

Algorithm

```
Step 1: Start
Step 2: Input the row and column size of matrix
Step 3: Input elements of matrix row-wise
Step 4: Initialize two variables to 0 to hold smallest -min and largest- max elements of the matrix
Step 5: Initialize 4 variables to hold row and column positions of smallest and largest elements
Step 6: Run a nested loop
        Step 6.1: Compare each element of matrix with min and max variables and replace it with
current element- Matrix[i][j]
                                max <---- Matrix[i][j] if Matrix[i][j] > max
                                min <---- Matrix[i][j] if Matrix[i][j] < min
        Step 6.2: Update row and column positions of smallest and largest elements.
Step 7: Print the smallest and largest number along with its position.
```

Program Code

```
def Matrix_input():
    """ Takes in a 2-D Matrix from user and returns array with row and colum specification"""

    Matrix_rows = int(input('Input the row size of Matrix '))
    Matrix_colums = int(input('Input the column size of Matrix '))
    print('\nEnter the elements row-wise ')
    Matrix = []

    for i in range(Matrix_rows):
```

```
        row_temp = []
        for j in range(Matrix_colums):
            row_temp.append(int(input()))
        Matrix.append(row_temp)
    print(f'Matrix =\n{Matrix}')
    return Matrix , Matrix_rows, Matrix_colums

Matrix, Matrix_rows, Matrix_colums = Matrix_input()
max = min = Matrix[0][0]
max_row = max_column = min_row= min_column = 0

# loop to mark the smallest and largest number in the matrix along with its row and column position

for i in range(Matrix_rows):
        for j in range(Matrix_colums):
            if(Matrix[i][j] > max):
                max = Matrix[i][j]
                max_row = i + 1
                max_column = j + 1

            if(Matrix[i][j] < min):
                min = Matrix[i][j]
                min_row = i + 1
                min_column = j + 1

print(f'The largest element is {max} at row {max_row} and column {max_column}')
print(f'The smallest element is {min} at row {min_row} and column {min_column}')
```

Result

```
Input the row size of Matrix 4

Input the column size of Matrix 3

Enter the elements row-wise

4

2

4

5

6

1

4

8

6

5

9
```

```
4
Matrix =
[[4, 2, 4], [5, 6, 1], [4, 8, 6], [5, 9, 4]]
The largest element is 9 at row 4 and column 2
The smallest element is 1 at row 2 and column 3
```

Question 1(b): Input two arrays and output the common values between the two.

Aim

Finding the non-repeated common elements between two 1-D arrays

Discussion

A nested loop is run to compare each element of array 1 to array 2 to find the common values between the two.

Algorithm

```
Step 1: Start
Step 2: Input the size of array 1
Step 3: Input array 1
Step 4: Input size of array 2
Step 5: Input array 2
Step 6: Declare an empty list- common_element to hold the common elements
Step 7: Run a nested loop
        Step 7.1: Compare each element of array 1 with array 2 and append first occurrence
        common elements to common_element list

Step 8: Print array 1 and 2 along with the common elements between the two.
```

Program Code

```
def array_input():

    """ Takes in input of 2 1-D arrays and returns the arrays with respective array lengths"""
    array1_length = int(input('Input length 0f 1-D array 1- '))
    print('\nEnter elements for array 1 ')
    array1 = []


    for j in range(array1_length):
            array1.append(int(input()))

    array2_length = int(input('Input length 0f 1-D array 2- '))
    print('\nEnter elements for array 2 ')
    array2 = []


    for j in range(array2_length):
            array2.append(int(input()))
```

```
    return array1, array1_length,array2, array2_length

array1, array1_length,array2, array2_length = array_input()
common_element= []
print(f'Array 1 =\n{array1}')
print(f'Array 2 =\n{array2}')

    #Running loop to find common elements between two arrays and adding same to a new array if
    that element is not already present

for array1_element in array1 :
        if (array1_element in array2) and (array1_element not in common_element) :
            common_element.append(array1_element)
print(f'The common elements in array 1 and 2 are :\n{common_element}')
```

Result

```
Input length Of 1-D array 1- 7

Enter elements for array 1

2

3

3

6

7

8

3

Input length Of 1-D array 2- 5

Enter elements for array 2

3

6

9

2

3
Array 1 =
[2, 3, 3, 6, 7, 8, 3]
Array 2 =
[3, 6, 9, 2, 3]
The common elements in array 1 and 2 are :
[2, 3, 6]
```

Question 1(c): Do element wise addition, subtraction, multiplication and division with and without built-in functions.

Aim

Performing Element-wise operations on two 1-D arrays with and without inbuilt functions

Discussion

A loop is run to perform element-wise operations between two arrays at same index positions.

Algorithm

```
Step 1: Start
Step 2: Input the size of array 1
Step 3: Input array 1
Step 4: Input size of array 2
Step 5: Input array 2
Step 6: Declare an empty list to hold results of element wise addition, subtraction,
multiplication and division operations
Step 7: Pad the shorter array with trailing zeros
Step 8: Run a nested loop
        Step 8.1: Perform element-wise addition, subtraction, multiplication and division
        operations and append it to their respective lists declared in Step 6.
Step 9: Print results of element wise operations without inbuilt function.
Step 10: Print results of element wise operations with inbuilt function.
```

Program Code

```
import numpy as np

def array_input():

    """ Takes in input of 2 1-D arrays and returns the arrays with respective array lengths"""
    array1_length = int(input('Input length 0f 1-D array 1 '))
    print('\nEnter elements for array 1 ')
    array1 = []


    for j in range(array1_length):
            array1.append(int(input()))

    array2_length = int(input('Input length 0f 1-D array 2 '))
    print('\nEnter elements for array 2 ')
    array2 = []


    for j in range(array2_length):
            array2.append(int(input()))


    return array1, array1_length,array2, array2_length

array1, array1_length,array2, array2_length = array_input()
```

```
Add = []
Sub = []
Mul = []
Div = []

if ( array1_length > array2_length):
        array1, array2 = array2, array1
        array1_length, array2_length = array2_length, array1_length

    # Appending the smaller array with trailing zeros for easier computation

for i in range(array2_length - array1_length):
        array1.append(0)

print(f'Array 1 =\n{array1}')
print(f'Array 2 =\n{array2}')

    # Running the loop to perform element-wise operations
for i in range(array2_length):

        Add.append(array1[i] + array2[i])
        Sub.append(array1[i] - array2[i])
        Mul.append(array1[i] * array2[i])
        Div.append(array1[i] / array2[i])


print('The element-wise operations on array 1 and array 2 without inbuit function are :')
print(f'Addition-\n{Add}')
print(f'Subtraction-\n{Sub}')
print(f'Division-\n{Div}')
print(f'Multiplication-\n{Mul}')

array1_np = np.array(array1)
array2_np = np.array(array2)

print('The element-wise operations on array 1 and array 2 without inbuit function are :')
print(f'Addition-\n{np.add(array1_np , array2_np)}')
print(f'Subtraction-\n{np.subtract(array1_np , array2_np)}')
print(f'Division-\n{np.divide(array1_np , array2_np)}')
print(f'Multiplication-\n{np.multiply(array1_np , array2_np)}')
```

Result

```
Input length Of 1-D array 1 7

Enter elements for array 1

3

5

6

8

9
```

```
3

5

Input length 0f 1-D array 2 4

Enter elements for array 2

6

9

8

7
Array 1 =
[6, 9, 8, 7, 0, 0, 0]
Array 2 =
[3, 5, 6, 8, 9, 3, 5]
The element-wise operations on array 1 and array 2 without inbuit function are :
Addition-
[9, 14, 14, 15, 9, 3, 5]
Subtraction-
[3, 4, 2, -1, -9, -3, -5]
Division-
[2.0, 1.8, 1.3333333333333333, 0.875, 0.0, 0.0, 0.0]
Multiplication-
[18, 45, 48, 56, 0, 0, 0]
The element-wise operations on array 1 and array 2 without inbuit function are :
Addition-
[ 9 14 14 15  9  3  5]
Subtraction-
[ 3  4  2 -1 -9 -3 -5]
Division-
[2.         1.8        1.33333333 0.875      0.         0.
 0.        ]
Multiplication-
[18 45 48 56  0  0  0]
```

Question 2: Read, display and save the "lenna.jpg" image to another format. Also display the image format, size, mode and information of the original image using built in commands.

Aim

Reading, displaying the image, displaying the information associated with the image using inbuilt functions and saving the image.

Discussion

An image can be read, shown, its associated information displayed and saved to another format using PIL library. PIL (Python Image Library) is used for the operations on an image file. It contains basic inbuilt functions for the same. Metadata of the image file can be retrieved by using ExifTags module in PIL. getexif()command is used to decoded the data into a human readable format.

Algorithm

```
Step 1: Start
Step 2: import Image and TAGS modules from PIL library
Step 3: Read image from image location
Step 4: Display Image
Step 5: Save the image into a different format ( here pdf )
Step 6: Display image format, size and mode using inbuilt function
Step 7: Get exif data of image
Step 8: Run a loop
        Step 8.1: For each tag id associated with image, decode and print the image metadata
```

Program Code

```
from PIL import Image
from PIL.ExifTags import TAGS

# Reading an image
imag_lenna = Image.open(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\Lenna.jpeg")

#Displaying the read image
imag_lenna.show()

#Saving the image as a pdf
lenna_pdf = imag_lenna.save(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\
Lenna_pdf.pdf")

print(f"The format of Lenna image is {imag_lenna.format}")
print(f"The size of Lenna image is {imag_lenna.size}")
print(f"The mode of Lenna image is {imag_lenna.mode}")

# Extracting metadata of image and displaying it

exifdata = imag_lenna.getexif()
# Iterating over all EXIF data fields

for tag_id in exifdata:

    tag = TAGS.get(tag_id, tag_id)
    data = exifdata.get(tag_id)

    try:
        if isinstance(data, bytes):
           data = data.decode()
    except :
        print()
    print(f"{tag:25}: {data}")
```

Result

```
The format of Lenna image is JPEG
The size of Lenna image is (512, 512)
The mode of Lenna image is RGB
ImageWidth               : 512
ImageLength              : 512
BitsPerSample            : (8, 8, 8)
```

```
Compression               : 1
PhotometricInterpretation: 2
ImageDescription          :
StripOffsets              :
```

(31, 10783, 21535, 32287, 43039, 53791, 64543, 75295, 86047, 96799, 107551, 118303, 129055, 139807, 150559, 161311, 172063, 182815, 193567, 204319, 215071, 225823, 236575, 247327, 258079, 268831, 279583, 290335, 301087, 311839, 322591, 333343, 344095, 354847, 365599, 376351, 387103, 397855, 408607, 419359, 430111, 440863, 451615, 462367, 473119, 483871, 494623, 505375, 516127, 526879, 537631, 548383, 559135, 569887, 580639, 591391, 602143, 612895, 623647, 634399, 645151, 655903, 666655, 677407, 688159, 698911, 709663, 720415, 731167, 741919, 752671, 763423, 774175, 784927)

```
Orientation               : 1
SamplesPerPixel           : 3
RowsPerStrip              : 7
StripByteCounts
```

: (10752, 10752, 10752, 10752, 10752, 10752, 10752, 10752, 10752,10752, 1536)

```
XResolution               : 28.34646
XPTitle                   :
YResolution               : 28.34646
```

Figure 1: Lenna.jpeg

Question 3: Familiarize the following basic commands in PIL: (a) crop, paste (b) split, merge (c) resize, rotate, transform (d) blend (e) convert, copy (f) getbands, getextrema, getpixel, putpixel

Aim

To familiarise crop, paste, split, merge, resize, rotate, transform, blend, convert, copy, getbands, getextrema, getpixel and putpixel commands in PIL.

Discussion

PIL (Python Image Library) is used for the operations on an image file. It contains basic inbuilt functions for the same.crop() and paste() functions enable cropping and pasting of image according to pixel position given. split() decomposes the image into bands and the merge() combines the bands into the original image.transform() command can be used to change size, add extra data and transform the image data

withthe specified method of transformation.blend() is used to blend two images of same size and mode with a parameter 'alpha' which specifies theopacity.By using convert() command, we can convert the mode of image(like RGB to L, 1, etc.)Bands of the image (mode) can be obtained by using getband() command. To find the extrema of pixelvalues, we can use getextrema() and if we wish to find the pixel values at a particular position, getpixel()command is used. putpixel() allows us to give arbitrary colour to a specified pixel (or group of pixels

Algorithm

```
Step 1: Start
Step 2: import Image module from PIL library
Step 3: Read image 'Lenna' and 'Cameraman' from image file location.
Step 4: Crop the image by giving top-left and bottom-right pixel location as crop area.
Step 5: Paste cropped image into another image by giving top-left pixel position at which to be
pasted.
Step 6: Split an image into R,B,G intensity arrays
Step 7: Merge back the split image ( RGB assembled as RBG )
Step 8: Resize image by giving resized width and height as parameters
Step 9: Rotate image by giving rotation angle in degrees
Step 10: Transform image using inbuilt function
Step 11: Blend two images of same size and mode.
Step 12: Convert the mode of image
Step 13: Copy an image.
Step 14: Print the bands of image.
Step 15: Print extreme values of image using inbuilt function
Step 16:  Print pixel intensity at a particular specified position
Step 17: Run a loop to insert pixels at a range of pixel positions
```

Program Code

```
from PIL import Image
Image_Cameraman = Image.open(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\
Cameraman.png")
Image_Lenna = Image.open(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\
Lenna.jpeg")
Image_Lenna_2 = Image.open(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\
Lenna.jpeg")

width, height = Image_Cameraman.size
width_1, height_1 = Image_Lenna.size

# a-1 ) Cropping the image from top-right pixel to bottom-left pixel position diagonally
imag_crop = Image_Cameraman.crop((0, height/4, width/2, 3*height/4))
imag_crop.show()
imag_crop.save(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\imag_crop.png")

# a-2 )Pasting cropped Cameraman image to Lenna image
Image.Image.paste(Image_Lenna, imag_crop, (20, 50))
Image_Lenna.show()
Image_Lenna.save(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\imag_paste.png")

# b-1 ) splitting image into red blue and green intensity array
imag_split  = Image.Image.split(Image_Lenna_2)
imag_split[0].show()
imag_split[0].save(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\imag_split_R.png")

imag_split[1].show()
```

```
imag_split[1].save(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\imag_split_G.png")

imag_split[2].show()
imag_split[2].save(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\imag_split_B.png")

# b-2 ) merging the image in the order RBG from split RGB intensity array
imag_merge = Image.merge( 'RGB', (imag_split[0], imag_split[2],imag_split[1]))
imag_merge.show()
imag_merge.save(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\imag_merge.png")

# c- 1 ) resizing an image
imag_resize = Image_Cameraman.resize((width -100,height - 100))
imag_resize.show()
imag_resize.save(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\imag_resize.png")

# c- 2 ) rotating an image
imag_rotate = Image_Cameraman.rotate(48)
imag_rotate.show()
imag_rotate.save(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\imag_rotate.png")

# c- 3 ) tranforming  an image to its extent at top-left (0-10) to bottom right (width/4, height/4))
imag_transform = Image_Cameraman.transform((width - 100, height-200), Image.EXTENT,
data =[10, 0, 10 + width_l // 4, height_l // 3 ])
imag_transform.show()
imag_transform.save(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\
imag_transform.png")

# d ) blending two images
blend_into_image = imag_transform.resize((width,height ))

blend_into_image.save(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\blend_into_imag
imag_blend = Image.blend(Image_Cameraman,blend_into_image, 25.0)

imag_blend.show()
imag_blend.save(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\imag_blend.png")

#e-1) Converting an image to diether mode 1 (noise inclusion)
imag_convert = Image_Cameraman.convert("1")
imag_convert.show()
imag_convert.save(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\imag_convert.png")

#e-2) Copying an image
imag_copy = Image_Cameraman.copy()
imag_copy.show()
imag_copy.save(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\
imag_copy.png")

#f-1) Printing the bands of an image
imag_band = Image_Cameraman.getbands()
print("The bands of image Cameraman are ", imag_band)

#f-2) Printing max and min pixel values of an image
imag_extremes = Image.Image.getextrema(Image_Cameraman)
print("The extreme(max and min) of image Cameraman are ", imag_extremes)

#f-3 ) Outputting pixel intensity at certain position
imag_getpixel = Image_Cameraman.getpixel((width//2, height//2))
print("The pixel intensity at ",(width//2, height//2),"are", imag_getpixel )
```

```
#f-4) Putting an range of pixels in an image

for i in range(height//4, height*3//4) :
    for j in range(width//4,width*3//4):
        imag_copy.putpixel((i,j),(200,125,300,255))

imag_copy.show()
imag_copy.save(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\imag_put_pixel.png")
```

Result

Figure 2: Cameraman : (a)Original (b)Cropped (c)Pasted

Figure 3: Cameraman : (a)Split-R (b)Split-G(c)Split-B d) Merge- RBG

Figure 4: Cameraman : (a)Resize (b)Rotate(c)Transform

Figure 5: Cameraman : (a)Image 1 (b)Image 2 (c)Blended Image

.
.
.
.
.

Figure 6: Cameraman : (a)Diether Mode 1 Image (b)Original (c) Copy

.
.
.
.

```
The bands of image Cameraman are  ('R', 'G', 'B', 'A')
The extreme(max and min) of image Cameraman are  ((0, 255), (0, 255), (0, 255), (255, 255))
The pixel intensity at  (128, 128) are (17, 17, 17, 255)
```

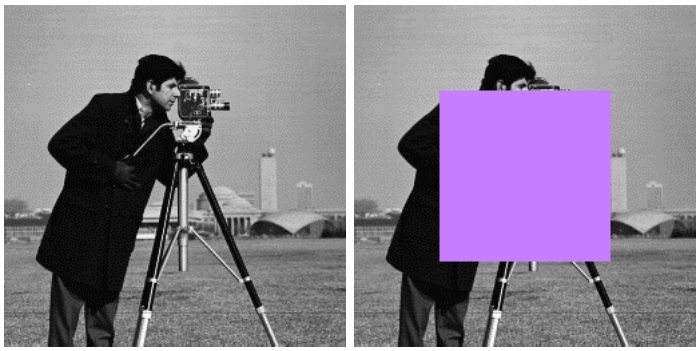

Figure 7: Cameraman : (a)Original (b) Pixel inserted

Question 4: Create, plot and save the Image (a) Fully White Image(HINT pixel values are 255) (b) Fully black Image(HINT pixel values are 0)

Aim

Creating and plotting a fully white and black image

Discussion

Fully black image corresponds to pixel intensity 0 and fully white image corresponds to pixel intensity 255. An array can be converted to an image object using Image.fromarray() module.

Algorithm

```
Step 1: Start
```

```
Step 2: Import Image module from PIL library and numpy library
Step 3: Input height and width of image to be created
Step 4: Create fully black and white array of respective pixel intensities.
Step 5: Convert arrays into Image objects
Step 6: Plot the fully black and white image
```

Program Code

```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt

#from IPython import get_ipython
#get_ipython().run_line_magic('matplotlib', 'inline')

image1_rows = int(input('Enter the height of the image to be created\n'))
image1_columns = int(input('Enter the width of the image to be created\n'))

image_white = np.full((image1_rows, image1_columns),255)
image_black = np.full((image1_rows, image1_columns),0)

image_white = np.uint8(image_white)
image_black = np.uint8(image_black)

image_white = Image.fromarray(image_white)
image_black = Image.fromarray(image_black)

fig = plt.figure()

fig.add_subplot(2,1,1)
plt.imshow(image_white, cmap= plt.get_cmap('gray'),vmin=0,vmax=255)


fig.add_subplot(2,1,2)
plt.imshow(image_black,  cmap= plt.get_cmap('gray'),vmin=0,vmax=255)


White_image = image_white.save(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\
White_Image.jpeg")
Black_image = image_black.save(r"C:\Users\Aruna Shaju K\Documents\Image-Processing-Lab-Images\
Black_Image.jpeg")
```

Result

```
Enter the height of the image to be created
600

Enter the width of the image to be created
700
```

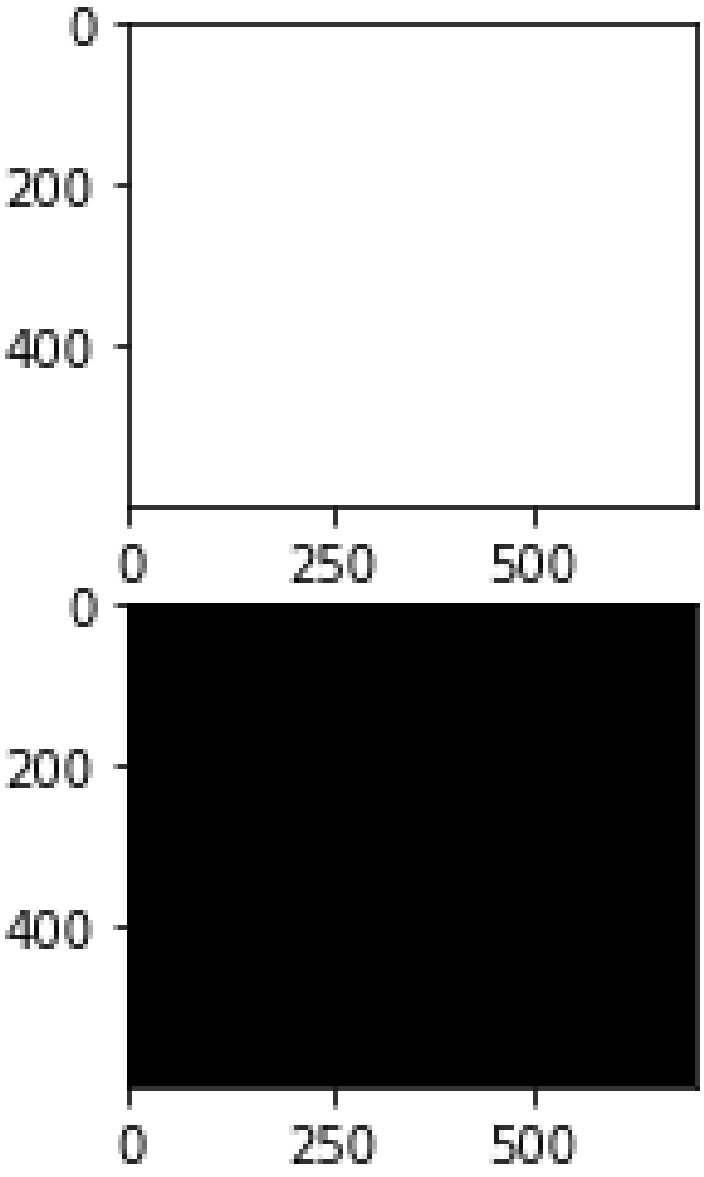

Figure 8: Fully White and Black Image Plot