HOWEWORK ASSIGNMENT 0

Problem 2.

1.

```
##### Importing all the libraries needed  #####
import cv2
import numpy as np
import matplotlib.pyplot as plt
import ipdb
#####################################################################
```

2.

2.a

```
########## section 2.a <Loading and displaying the image using cv2.imshow
>#######################
#load image
img = cv2.imread('elephant.jpeg')
#display image
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
####################################
```

2.b

########## section 2.b <displaying the image using Matplotlib >#########################

#display image

plt.imshow(img)

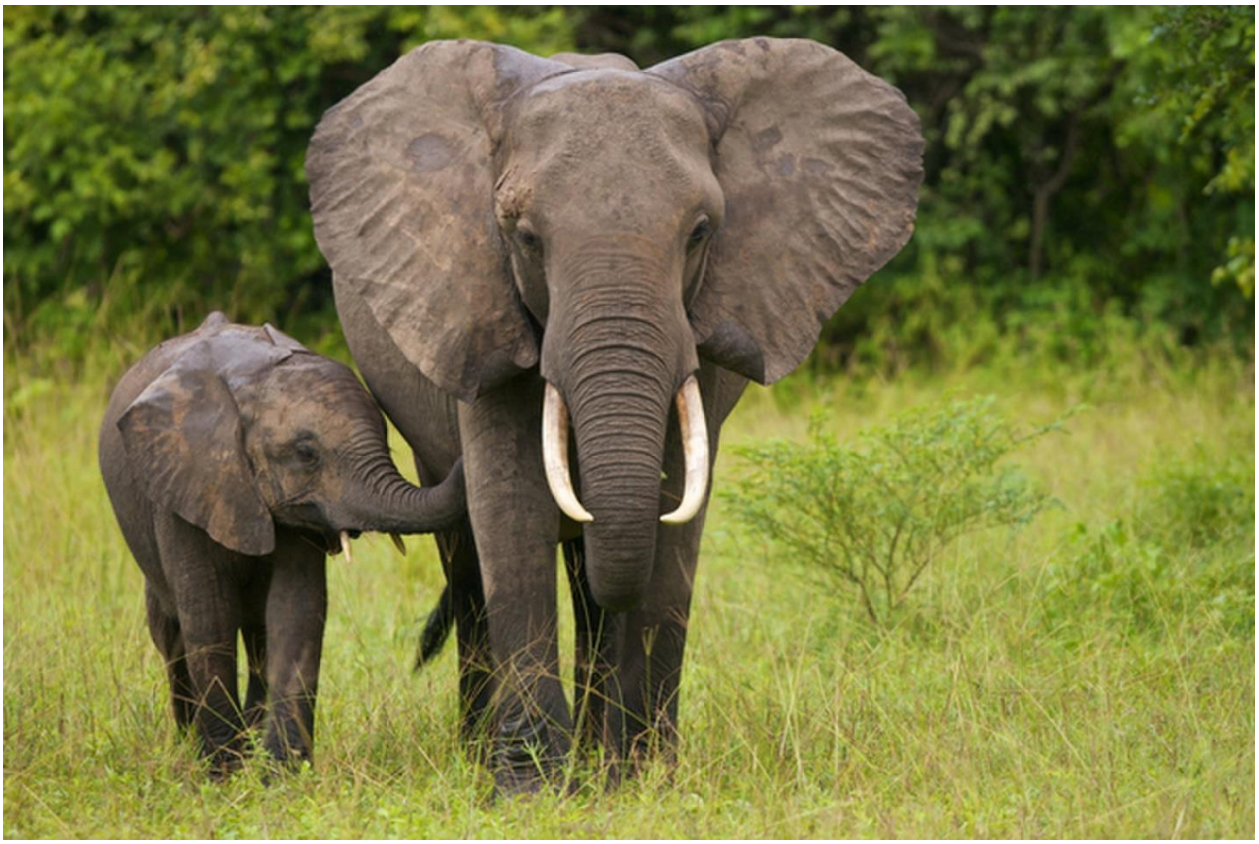
plt.show()

#save image

cv2.imwrite('elephant_opencv.png', img)

#####################################

Output Image:



2.c

########## section 2.c <converting bgr to rgb >#######################

#converting bgr to rgb format

imgaa = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

display image

plt.imshow(imgaa)

plt.show()

#save image

cv2.imwrite('elephant_matplotlib.png', imgaa)

####################################

Output Image:

Color image loaded by OpenCV is in BGR mode. But Matplotlib displays in RGB mode. So color images will not be displayed correctly in Matplotlib if image is read in OpenCV.

2.d

########## section 2.d <read image in gray scale >#######################

#converting image to gray

gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

display image

plt.imshow(gray_img, cmap ='gray')

plt.show()

save image

cv2.imwrite('elephant_gray.png', gray_img)

####################################

Output image:

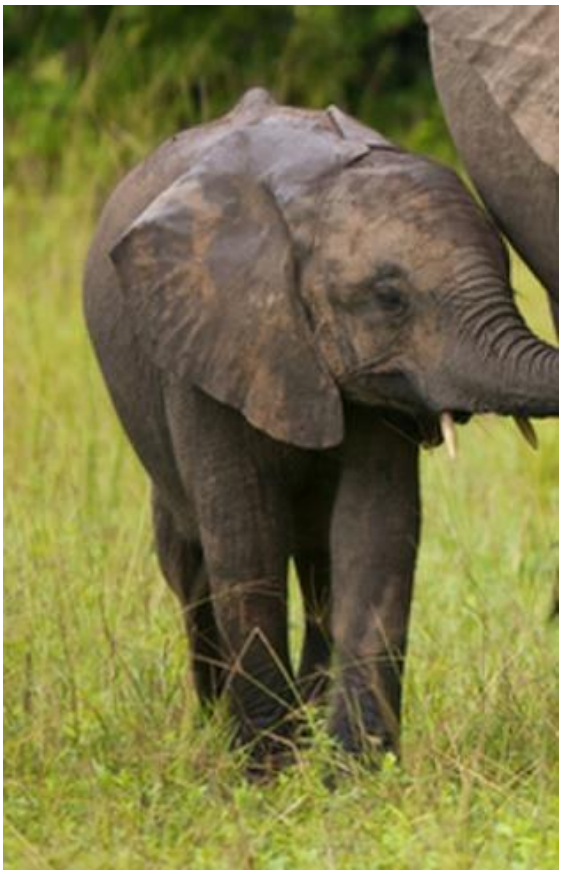
3.

########## section 3 <cropping the image >#######################

```
# crop image
crop_img = img[300:1000, 50:500]
# cropped image is converted to rbg to display it using plt.imshow
imga1 = cv2.cvtColor(crop_img, cv2.COLOR_BGR2RGB)
# display image
plt.imshow(imga1)
plt.show()
# save image. We use the bgr format input for cv2.imwrite
cv2.imwrite('babyelephant.png', crop_img)
```

####################################

Output Image:



4.

4.a

```
########## section 4.a <read image in color and convert to rgb >#######################
#load image
img1 = cv2.imread('elephant.jpeg')
#converting to rgb space
img2 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
#####################################
```

4.b

```
########## section 4.b <adding 256 to every pixel using numpy >#######################
img3 = img2 + 256
print('image dtype ',img3.dtype)
img4 = np.uint8(img3)
print('image dtype',img4.dtype)
plt.imshow(img4)
plt.show()
#####################################
```

Pixel values donot change as numpy uses modulo operation on overflow. i.e as example, 1 + 256 = 257. 257/256 = 1. So, we get the same pixel value again.

4.c

```
########## section 4.c <adding 256 to every pixel using openCV >#######################
b,g,r = cv2.split(img1)
k1 =cv2.add(b,256)
k2 =cv2.add(g,256)
k3 = cv2.add(r,256)
# the image is merged in rgb format for displaying the image using plt.imshow
img22 = cv2.merge((k3,k2,k1))
plt.imshow(img22)
plt.show()
```

####################################

OpenCV’s add vs Numpy’s add:

NumPy uses "modulo" arithmetic on overflow whereas openCV uses clipping on overflow. As 256 is added to all the pixels, it reaches saturation and gives the output image with all pixels white.

5.

5.a

########## section 5.a <reading image and converting to rgb space>#######################

#load image

img1b = cv2.imread('elephant.jpeg')

#converting to rgb space

img2b = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

####################################

5.b

########## section 5.b <downsample image by factor 10>#######################

img_scaled = cv2.resize(img1b,None,fx=0.1, fy=0.1)

the image is converted to rgb format for displaying the image using plt.imshow

img2bb = cv2.cvtColor(img_scaled, cv2.COLOR_BGR2RGB)

plt.imshow(img2bb)

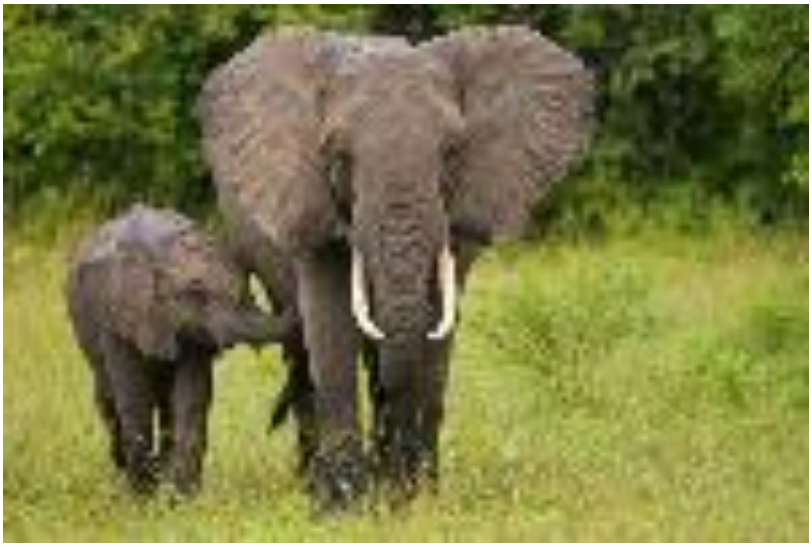
plt.show()

the image in bgr format is saved using imwrite

cv2.imwrite('elephant_10xdown.png', img_scaled)

####################################

Output Image:



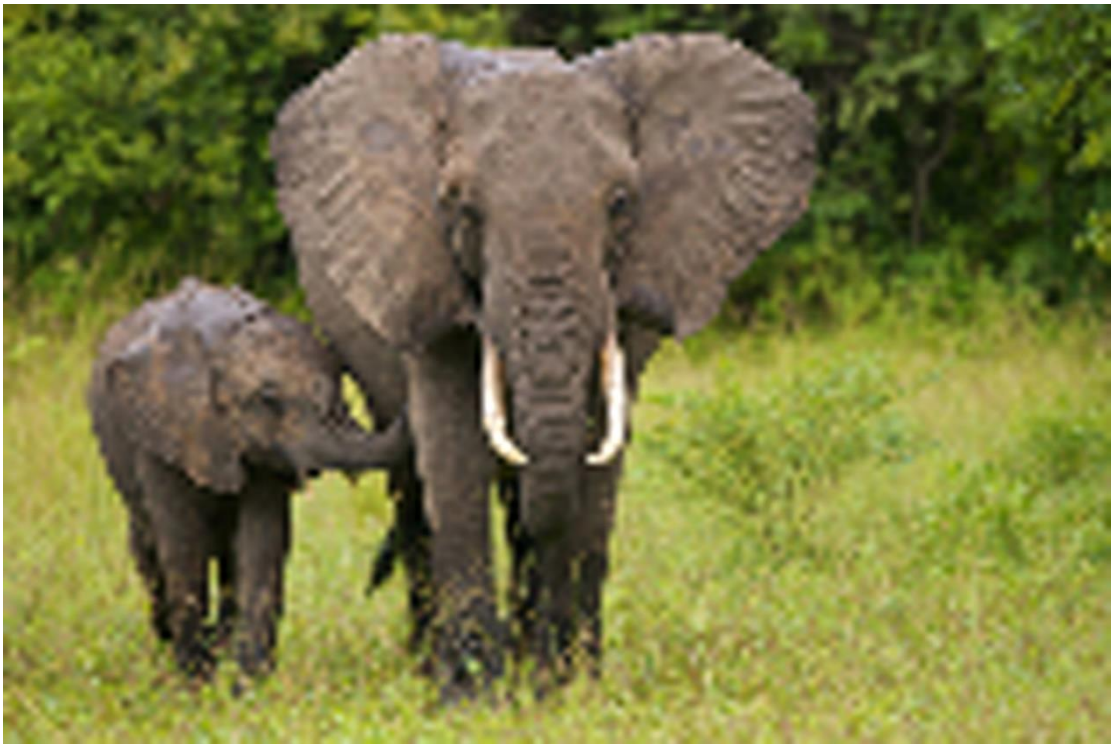
5.c

```
########## section 5.c <upsample image by factor 10 using 2 methods>#######################
#upsampling by BICUBIC method
img_scaled1 = cv2.resize(img_scaled,None,fx=10, fy=10, interpolation = cv2.INTER_CUBIC)
# the image is converted to rgb format for displaying the image using plt.imshow
imgb2 = cv2.cvtColor(img_scaled1, cv2.COLOR_BGR2RGB)
plt.imshow(imgb2)
plt.show()
# the image in bgr format is saved using imwrite
cv2.imwrite('elephant_10xup_bicubic.png', img_scaled1)
#upsampling by NEAREST NEIGHBOUR
img_scaled2 = cv2.resize(img_scaled,None,fx=10, fy=10, interpolation = cv2.INTER_NEAREST)
# the image is converted to rgb format for displaying the image using plt.imshow
imgb21 = cv2.cvtColor(img_scaled2, cv2.COLOR_BGR2RGB)
plt.imshow(imgb21)
plt.show()
# the image in bgr format is saved using imwrite
cv2.imwrite('elephant_10xup_nearestneighbour.png', img_scaled2)
```

####################################

Ouput Images:

1. Upsampled using Bicubic



2. Upsampled using Nearest neighbor

5.d

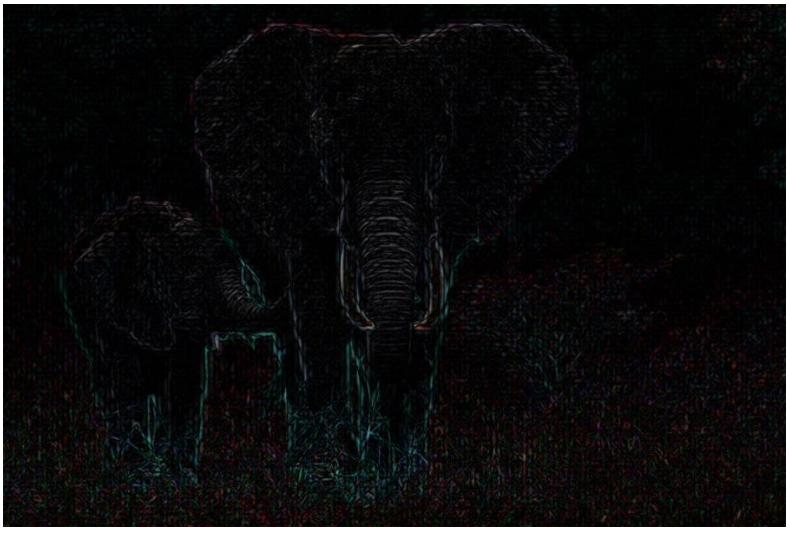
########## section 5.d <absolute difference between groud truth and up sampled images>#######################

```
# diff image for bicubic

imdiff1 = cv2.absdiff(img1b,img_scaled1)

cv2.imwrite('elephant_diffimg_bicubic.png', imdiff1)

# diff image for nearest neighbour

imdiff2 = cv2.absdiff(img1b,img_scaled2)

cv2.imwrite('elephant_diffimg_nearestneighbour.png', imdiff2)

# sum of all pixels in diff image for bicubic

k = cv2.sumElems(imdiff1)

k1 = sum(k)

print(k1)

# sum of all pixels in diff image for nearest neighbour

k2 = cv2.sumElems(imdiff2)

k3 = sum(k2)

print(k3)
```

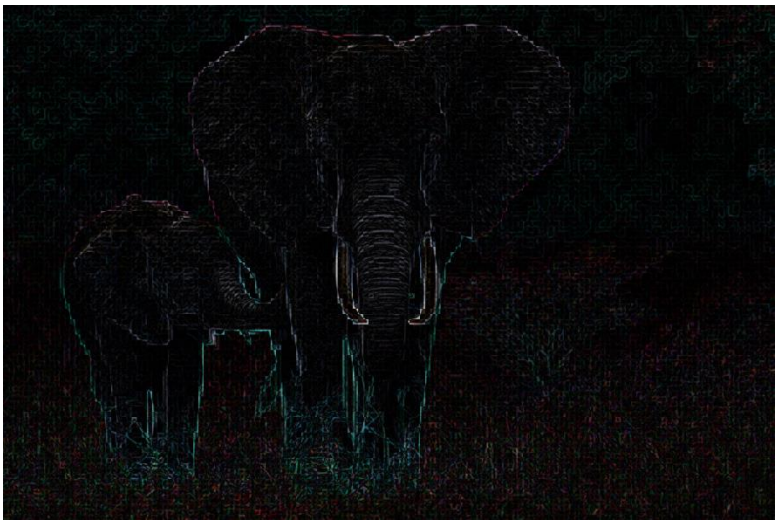
####################################

Output Images:

1. Difference image using bi cubic



2. Difference image using Nearest neighbor



Sum of all pixels in the difference image for bicubic method : 40260972.0

Sum of all pixels in the difference image for nearest neighbor method : 46599470.0

So, Bicubic method caused less error in unsampling for the image provided.

Code:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import ipdb

########## section 2.a <Loading and displaying the image >#######################
#load image
img = cv2.imread('elephant.jpeg')
#display image
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
#####################################

########## section 2.b <displaying the image using Matplotlib >#######################
#display image
plt.imshow(img)
plt.show()
#save image
cv2.imwrite('elephant_opencv.png', img)
#####################################

########## section 2.c <Loading and displaying the image >#######################
#converting bgr to rgb format
imgaa = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# display image
plt.imshow(imgaa)
plt.show()
```

```
#save image
cv2.imwrite('elephant_matplotlib.png', imgaa)
####################################

########## section 2.d <read image in gray scale >#######################
#converting image to gray
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# display image
plt.imshow(gray_img, cmap ='gray')
plt.show()
# save image
cv2.imwrite('elephant_gray.png', gray_img)
####################################

########## section 3 <cropping the image >#######################
# crop image
crop_img = img[300:1000, 50:500]
# display image
imga1 = cv2.cvtColor(crop_img, cv2.COLOR_BGR2RGB)
plt.imshow(imga1)
plt.show()
# save image
cv2.imwrite('babyelephant.png', crop_img)
####################################

########## section 4.a <read image in color and convert to rgb >#######################
#load image
img1 = cv2.imread('elephant.jpeg')
#converting to rgb space
```

```
img2 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
#####################################

########## section 4.b <adding 256 to every pixel using numpy >#######################
img3 = img2 + 256
print('image dtype ',img3.dtype)
img4 = np.uint8(img3)
print('image dtype',img4.dtype)
plt.imshow(img4)
plt.show()
#####################################

########## section 4.c <adding 256 to every pixel using opencv >#######################
b,g,r = cv2.split(img1)
k1 =cv2.add(b,256)
k2 =cv2.add(g,256)
k3 = cv2.add(r,256)
img22 = cv2.merge((k3,k2,k1))
plt.imshow(img22)
plt.show()
#####################################

########## section 5.a <reading image and converting to rgb space>#######################
#load image
img1b = cv2.imread('elephant.jpeg')
#converting to rgb space
img2b = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
#####################################
```

```
########## section 5.b <downsample image by factor 10>#######################
img_scaled = cv2.resize(img1b,None,fx=0.1, fy=0.1)
img2bb = cv2.cvtColor(img_scaled, cv2.COLOR_BGR2RGB)
plt.imshow(img2bb)
plt.show()
cv2.imwrite('elephant_10xdown.png', img_scaled)
#####################################

########## section 5.c <upsample image by factor 10 using 2 methods>#######################
#upsampling by BICUBIC method
img_scaled1 = cv2.resize(img_scaled,None,fx=10, fy=10, interpolation = cv2.INTER_CUBIC)
imgb2 = cv2.cvtColor(img_scaled1, cv2.COLOR_BGR2RGB)
plt.imshow(imgb2)
plt.show()
cv2.imwrite('elephant_10xup_bicubic.png', img_scaled1)
#upsampling by NEAREST NEIGHBOUR
img_scaled2 = cv2.resize(img_scaled,None,fx=10, fy=10, interpolation = cv2.INTER_NEAREST)
imgb21 = cv2.cvtColor(img_scaled2, cv2.COLOR_BGR2RGB)
plt.imshow(imgb21)
plt.show()
cv2.imwrite('elephant_10xup_nearestneighbour.png', img_scaled2)
#####################################

########## section 5.d <absolute difference between groud truth and up sampled
images>#######################
# diff image for bicubic
imdiff1 = cv2.absdiff(img1b,img_scaled1)
cv2.imwrite('elephant_diffimg_bicubic.png', imdiff1)
# diff image for nearest neighbour
```

```
imdiff2 = cv2.absdiff(img1b,img_scaled2)
cv2.imwrite('elephant_diffimg_nearestneighbour.png', imdiff2)
# sum of all pixels in diff image for bicubic
k = cv2.sumElems(imdiff1)
k1 = sum(k)
print(k1)
# sum of all pixels in diff image for nearest neighbour
k2 = cv2.sumElems(imdiff2)
k3 = sum(k2)
print(k3)
#####################################
```