Need to fix approach. Split up image into 8x8 blocks and randomly pick out N pixels from each block.

MP1

February 3, 2022

1 Mini Project 1

1.0.1 Luis Pereda

[1]:
```
# This is for ECE580: Intro to machine learning Spring 2020 in Duke
# This is translated to Python from show_chanWeights.m file provided by Prof. Li by 580 TAs

# import ext libs
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# from scipy.misc import imread   # Make Sure you install the required packages like Pillow and scipy

def imgRead(fileName):
    """
    load the input image into a matrix
    :param fileName: name of the input file
    :return: a matrix of the input image
    Examples: imgIn = imgRead('lena.bmp')
    """
    imgIn = plt.imread(fileName)
    return imgIn


def imgShow(imgOut):
    """
    show the image saved in a matrix
    :param imgOut: a matrix containing the image to show
    :return: None
    """
    imgOut = np.uint8(imgOut)
    plt.imshow(imgOut)


def imgRecover(imgIn, blkSize, numSample):
    """
```

CAN ADD cmap = 'grey' to change image colors

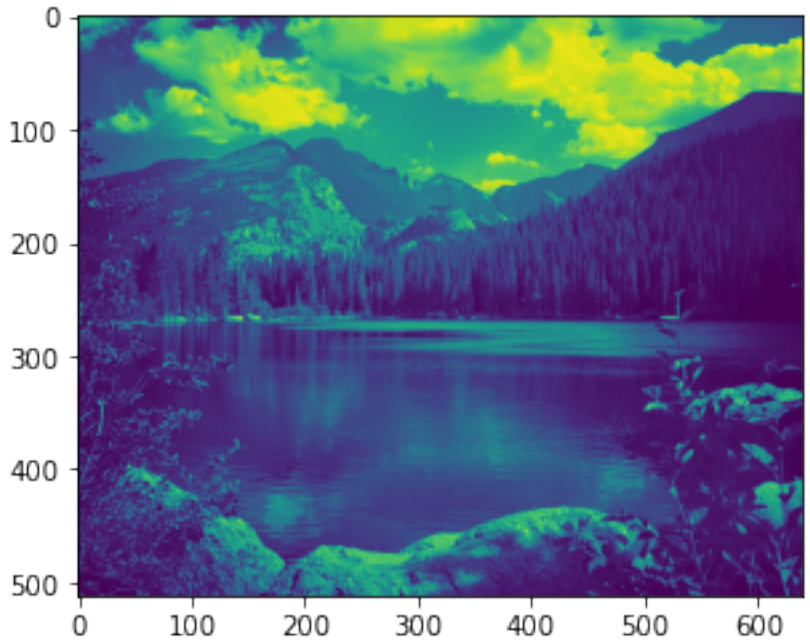
```
    Recover the input image from a small size samples
    :param imgIn: input image
    :param blkSize: block size
    :param numSample: how many samples in each block
    :return: recovered image
    """
    ##### Your Implementation here

    return None

"""
if __name__ == '__main__':
    a = imgRead('lena.bmp')
    print(np.shape(a))
    imgShow(a)
    print(a)
"""
```

[1]:
```
"\nif __name__ == '__main__':\n    a = imgRead('lena.bmp')\n
print(np.shape(a))\n    imgShow(a)\n    print(a)\n"
```

[21]:
```
fishing_boat = imgRead("fishing_boat.bmp")
```

[22]:
```
nature = imgRead("nature.bmp")
```

[23]:
```
imgShow(nature)
```



[55]:
```
(fishing_boat[:1, :4])
```

[55]:
```
array([[176, 177, 178, 179]], dtype=uint8)
```

[56]:
```
fishing_boat
```

[56]:
```
array([[176, 177, 178, …, 182, 160, 125],
       [177, 176, 177, …, 170, 131,  88],
       [176, 176, 176, …, 145,  98,  52],
       …,
       [100,  95,  99, …,  11,  14,  16],
       [ 98,  94, 103, …,  10,   8,   3],
       [ 64,  64,  64, …,  71,  71,  71]], dtype=uint8)
```

[73]:
```
sampled_image = []

count1 = 0
for row in fishing_boat:
    count = 0
    sampled_row = []
    for col in row:
        if (count % 4 == 0):
            sampled_row.append(col)
        count += 1
    if (count1 % 4 == 0):
        sampled_image.append(sampled_row)
    count1 += 1
```

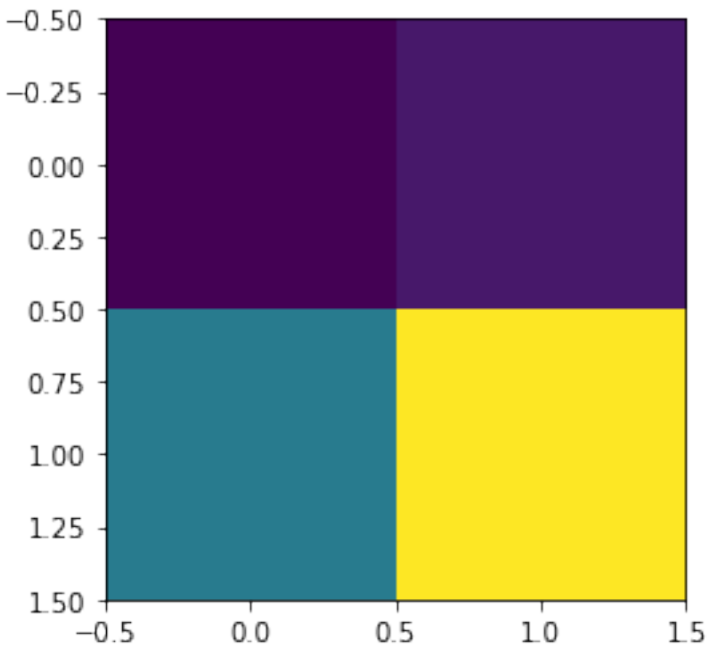
[85]:
```
def sample_image(image, k):
    sample = []
    for i in range(0, len(image), k):
        sample_row = []
        for j in range(0, len(image[0]), k):
            sample_row.append(image[i][j])
        sample.append(sample_row)
    return sample
```

CAN REPLACE THIS WHOLE METHOD WITH IMAGE[::N, ::M]

Ends up being only one line

[88]:
```
imgShow(sample_image(fishing_boat, 100))
```

Check to see that image is divisible by step size (sample by every 30).


−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50
−0.5
0.0
0.5
1.0
1.5

[ ]: