Homework 5 Notes

- Start by making sure you watched and understood the lecture from Friday 3/4/2022. The homework guides you through a 2D version of the signal recovery process presented for a 1D signal.

1. Image Compression using DCT on "The Son of Man"
    a. Load image. Downsample to 53x41 pixels and convert to grayscale
        i. You can use built in functions from scikit image, ndimage, or cv2
        ii. You should include what this downsampled/grayscale image looks like in your report since you will be assessing the quality of low dimensional reconstructions compared to this original.
    b. Take DCT
        i. Matlab: Take the discrete cosine transform of image using dct2(im_gray)
        ii. Python:
            1. Construct D using function construct_D_mat() provided in helper code
            2. Vectorize image using im_gray.flatten()
            3. Matrix-vector multiply D and vectorized version of image to get DCT. Remember to use np.dot() or @, not just *
    c. Make a plot of DCT. These are the coefficients needed to construct the image from cosines of different frequencies. [This plot should definitely be included in your report]
    d. Threshold to the first 5, 10, 20, 40% of the coefficients:
        i. Take the absolute value and sort coefficients by size. Identify the magnitude of the smallest coefficient that makes it into the top 5% or 10% or 20% etc. This will be your threshold on the magnitude of the coefficients. You might want to indicate these values on your plot.
        ii. In the original DCT, set the coefficients that have a magnitude (absolute value) below the threshold value to zero in the DCT. This is your thresholded DCT
        iii. Reconstruct image by multiplying D^{-1} by the thresholded DCT, where D^{-1} is constructed using the construct_iD_mat() provided. In matlab you can use idct2().
        iv. How do the reconstructions look? Which features are well-captured? What noise is introduced? How would you assess the compressibility of the image?
        v. You should make a figure that shows the reconstructed images for each threshold. When displaying images it is a good idea to turn the axes off. You probably want to include some titles, in which case you will need to increase the font size. Here is some example python code to make a figure with 4 subplots, showing the images in the list `Reconstructions`.

```
matplotlib.rcParams.update({'font.size': 18})
Reconstructions = [img_5, img10, img_20, img_40]
Titles = ['5%', '10%', '20%', '40%']
fig, ax = plt.subplots(2, 2, figsize=(20,20))
count = 0
for j in range(0,2):
      for jj in range(0,2):
            ax[j,jj] = plt.imshow(Reconstructions[count])
            a[j,jj].axis('off')
            ax[j,jj].set_title(Titles[count])
            count = count + 1
```

2. Compressed Image Recovery on the Son of Man
    a. Remember we are still working with the same downsampled, grayscale image.
    b. Construct a "measurement matrix", B, as described in the HW. The dimensions of B will be MxN where M is the number of points measured, and N is the number of pixels in the image. Each row of B has a 1 in the column that corresponds to the pixel which is being sampled, so this will be a sparse matrix! If you constructed a mask by setting the indices of the original image which correspond to columns containing non-zero entries in B to 0, it would look like salt and pepper noise. Check out `plt.imshow(np.reshape(np.sum(B, axis = 0), (Ny, Nx)), cmap = 'gray')`
    The white entries show which pixels you are measuring. You could multiply this element-wise by the image to see which pixels you are measuring and display black squares in place of the excluded pixels.
    c. Generate your measurements by matrix-vector multiplying B with the flattened image.
    d. Follow step 2c on the homework carefully. Remember to use matrix multiplication and use the hint about settings for cvx. Look at the lecture code for how to set up problems in cvx. You will define the variable you are minimizing, `x`, the objective function, and the constraint. It takes ~2 minutes to converge when I run the code on colab for M = 0.6*N. After the problem is solved, you can access x* by using `x.value`
    e. Recall that x* is the DCT of your reconstructed image. To see how the image reconstruction performed, reshape the product of D^{-1} times x* to the image dimensions.
    f. Again, follow step 2d carefully. See note above for creating figures with subplots once you are ready to make a figure for your report. For task 2 on the assignment, this figure is your main result. The prior steps of the task are helping you get set up for this, and don't necessarily need a separate figure/table. If you have space though, you might want to visualize an example of what the pixel measurements look like for one figure in your algorithm implementation section.
3. Unknown image reconstruction.

a. The measurements of the image are in y and the measurement matrix B tells us where the measurements were taken. This time, the measurements are not binary, but rather weights assigned to each pixel.
b. Use the process you developed in task 2 to reconstruct the image. The measurements are ‘y’. The measurement matrix is ’B’. The dimensions are Nx = 50 and Ny =50, so you will need to construct the appropriately sized D and D^{-1}.
c. Make sure to include your reconstructed image in the report!