EAS 596, Fall 2018, Homework 8
Due Thurs. 10/25, **9 AM**, Box outside Furnas 611

Work all problems. Only MATLAB code will be accepted. Show all work, including any M-files you have written or adapted. Make sure your work is clear and readable - if the TA cannot read what you've written, he will not grade it. All electronic work (m-files, etc.) **must** be submitted through UBLearns and submitted by the due time. Any handwritten work may be submitted by the due time. Each problem will be graded according to the following scheme: 2 points if the solution is complete and correct, 1 point if the solution is incorrect or incomplete but was using correct ideas, and 0 points if using incorrect ideas.

1. Write a MATLAB program that sets up a 15×40 matrix with entries zero everywhere except for values 1 in the positions indicated in Figure 1. The upper-leftmost 1 is in the (2,2) position while the lower-rightmost 1 is in the (13,39) position. The picture was produced using the `spy(A)` command.

   (a) Call `svd` to compute the singular values of A and print the results. Plot these numbers using both `plot` and `semilogy`. What is the exact rank of A?

   (b) For each i from 1 to rank(A), construct the rank-i matrix B that is the best approximation to A in the 2-norm. Use the command `pcolor(B)` with `colormap(gray)` to create images of these various approximations.

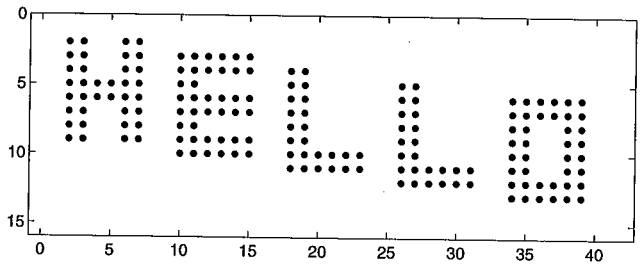

Figure 1: Figure generated by `spy(A)` command.

2. Bitmap images, such as jpg or png files, are nothing but matrices with a single number at each pixel location indicating the intensity of a

particular color. If the image is an 8-bit gray-scale image then one single matrix will suffice, with a value of 0 indicating black and 255 indicating white. Values between 0 and 255 indicate how gray the color at that pixel is to be. Color images are typically composed of three such matrices: one for red, one for green, and one for blue. Using the intensity of each of these three colors allows for other colors, such as purple, to be shown at a pixel. As with the gray-scale images the scalar values for each of these colors ranges from 0 to 255 for 8-bit images.

Recall from class that any matrix can be decomposed into the summation of rank-1 matrices:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{u}_1\sigma_1\mathbf{v}_1^T + \mathbf{u}_2\sigma_2\mathbf{v}_2^T + \cdots + \mathbf{u}_r\sigma_r\mathbf{v}_r^T,$$

where σ is a singular value, $\mathbf{u}$ and $\mathbf{v}$ are singular vectors, and r is the rank of matrix $\mathbf{A}$.

To compress an image you simply truncate the sum at the p^{th} singular value:

$$\mathbf{A} \approx \mathbf{u}_1\sigma_1\mathbf{v}_1^T + \mathbf{u}_2\sigma_2\mathbf{v}_2^T + \cdots + \mathbf{u}_p\sigma_p\mathbf{v}_p^T,$$

where $p < r$. What this means is that instead of storing the entire image matrix, we only need to store a small number of vectors and their associated singular values to create an approximation of an image.

In this problem, we will use MATLAB to study image compression using the SVD.

(a) Write a MATLAB function that takes in a file name and the SVD rank approximation p. The function will use MATLAB's `imread` function to read in the image then use MATLAB's `svd` command to construct the SVD of the image. Note you'll need to convert the image array from integers to doubles using the `double` function. Also note you need to do each of the three color arrays separately. The function should return the compressed image as an $m \times n \times 3$ array (the same dimensions as the array generated by the `imread` function).

(b) Write a MATLAB function that takes in an input file name, an array of SVD rank approximations $\mathbf{p}$, and an output file name. The function should take the input filename and for each SVD approximation rank, call your function to get the array for the

compressed image of the given rank. Then, use the `imshow` command, together with the `subplot` command to show comparisons of different image compressions. Figure 2 below shows an example.

(c) Write a MATLAB script that uses the previous function to perform image compression on each of the 3 images included with the assignment: `square.png`, `UB.png`, and `futurama.png`. What is the minimum rank you feel gives a good quality image for each one? What is your reasoning? Show at least five different low-rank approximations for each image.

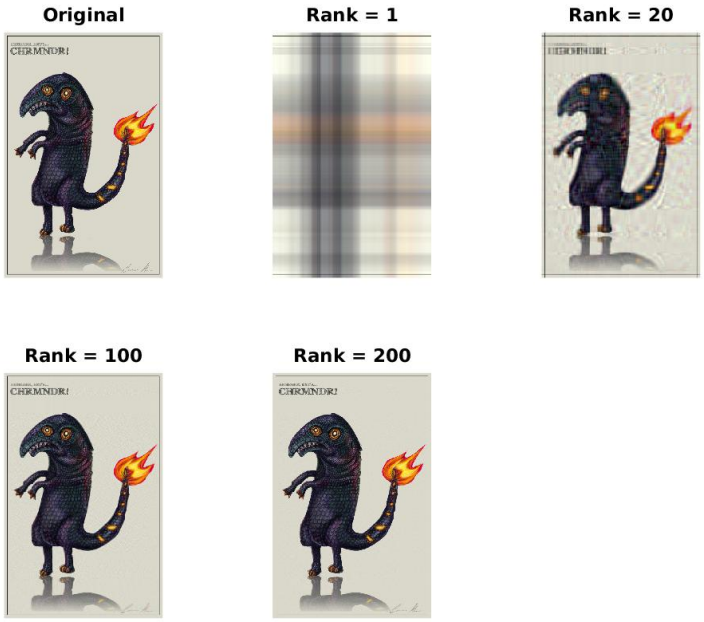

Figure 2: Example output of image compression comparsion.