Project: SVD image compression

Suppose $\boldsymbol{A}$ is a matrix of pixel intensity values representing an image. Recall that if $\boldsymbol{A} = \boldsymbol{USV}^T$ is an SVD, then we construct an approximation $\boldsymbol{A}_k = \boldsymbol{U}_k\boldsymbol{S}_k\boldsymbol{V}_k^T$ using the first k columns of $\boldsymbol{U}$ and $\boldsymbol{V}$, and the first k singular values only. This is the "best" approximation to $\boldsymbol{A}$ in a certain precise sense: among all matrices of rank k or less, $\|\boldsymbol{A} - \boldsymbol{A}_k\|$ has the smallest possible value. This optimality is true in the 2-norm, and in another norm that serves us better here, the Frobenius norm:

$$\|\boldsymbol{X}\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |X_{ij}|^2 \right)^{1/2}. \tag{1}$$

Unlike our other matrix norms, the Frobenius norm is not induced by a vector norm. However, in the image context there is a relationship: if $\boldsymbol{Z}$ is an image in matrix form and $\boldsymbol{z}$ is its vector form, then $\|\boldsymbol{Z}\|_F = \|\boldsymbol{z}\|_2$. The Frobenius norm is also closely related to the SVD of $\boldsymbol{A}$ and the formulation of $\boldsymbol{A}_k$, via

$$\begin{aligned} \|\boldsymbol{A}\|_F^2 &= \sigma_1^2 + \sigma_2^2 + \cdots + \sigma_n^2, \\ \|\boldsymbol{A}_k\|_F^2 &= \sigma_1^2 + \cdots + \sigma_k^2. \end{aligned} \tag{2}$$

The low-rank approximation $\boldsymbol{A}_k$ is not necessarily a great way to compress an image. However, it might be better if we break the image into blocks and apply compression to the blocks. That is your goal in this project.

1. Write a function `k = cutoff(sigma,p)`. Here `sigma` is a vector of n singular values and `p` is a number between 0 and 1. The value `k` that is returned should be the smallest possible k such that
$$\frac{\sigma_1^2 + \cdots + \sigma_k^2}{\sigma_1^2 + \cdots + \sigma_n^2} \ge p^2. \tag{3}$$

2. Write a function `[Z,ratio] = svdcompress(X,b,p)` that performs block SVD compression. $\boldsymbol{X}$ is divided into nonoverlapping $b \times b$ blocks. For example, the first block is `X(1:b,1:b)`. A block $\boldsymbol{A}$ should have its SVD computed. The singular values and p are passed to the `cutoff` function to determine a value of k for the block. The corresponding approximation $\boldsymbol{A}_k$ will occupy the same position in $\boldsymbol{Z}$ as $\boldsymbol{A}$ does in $\boldsymbol{X}$.

   The number of scalars needed to define $\boldsymbol{A}_k$ is $k(2b+1)$. Therefore, if $k \ge b/2$, you should use $\boldsymbol{A}$ rather than $\boldsymbol{A}_k$ in the output $\boldsymbol{Z}$, as it will be more efficient. In addition, along the bottom and right edges of the image there may be blocks whose width or height is smaller than b. Those can be copied directly into the output $\boldsymbol{Z}$.

   You also should keep a running total of the number of values needed to reconstruct $\boldsymbol{Z}$ (that is, either $k(2b + 1)$ or the number of elements in the block). At the end of the function, divide this total by the total number of elements in $\boldsymbol{X}$ to get the output value `ratio`, which represents the compression ratio achieved. It should be no greater than one in all cases.

3. Apply your `svdcompress` to the image you selected for the first project. Try block sizes 8, 16, 24, and 32 and a cutoff threshold $p = 0.5$. Display the resulting images and report the compression ratio in each case.

4. Apply `svdcompress` to your image, with the blocksize that you judge to give the best results in part 3, and cutoff thresholds of $p = 0.2$, 0.4, 0.6, and 0.8. Again, display the images and report the compression ratios.

For your submission, turn in your images and your functions.