Image Compression using SVD and EVD

HEMA LANDA EE19B036

30-01-2022

Motivation

We often need to transmit and store the images in many applications. Smaller the image, less is the cost associated with transmission and storage. So we often need to apply data compression techniques to reduce the storage space consumed by the image. Two approaches for this are to apply Singular Value Decomposition (SVD) and Eigen Value Decomposition(EVD) on the image matrix.In these methods, digital image is given to SVD/EVD. They refactors the given digital image into three matrices. Singular values or eigen values are used to refactor the image and at the end of this process, image is represented with smaller set of values, hence reducing the storage space required by the image.

Singular Value Decomposition

In this method the original matrix is represented by 3 matrices

$$A = U\Sigma V^T$$

Here, if the dimensions of matrix A is (n*m) then U is a (n*n) orthogonal matrix, Σ is a (n*m) diagonal matrix of singular values and V is (m*m) orthogonal matrix. These are calculated in following way. Note that original SVD form for any complex matrix deals with hermition forms but here since image has real matrix values we simply use transpose.

$$AA^TU = U\Sigma^2$$

$$V^T = \Sigma^{-1}U^{-1}A$$

If we see here, U is matrix of eigen vectors and Σ^2 is eigen value matrix of $A.A^T$ which we calculated using *np.linalg.eigh* since $A.A^T$ is symmetric matrix. From which we calculated V^T taking inverse of sigma and U.
From these 3 matrices, where they are properly arranged according to their weightage, we tried to reconstruct the original image by increasing number of modes(number of columns or value of k). For example for $k = 5$ we have taken first 5 columns of U matrix, top 5 sigma values and top 5 rows of V^T matrix.

```
approx_I = U[:,:k]@S[:k,:k]@VH[:k,:]
```

Eigen Value Decomposition

In this method also the original matrix is represented by 3 matrices

$$A = X\Lambda X^{-1}$$

Here Λ is eigen value matrix of A and X is eigen vector matrix. Again these are arranged according to their weightage. These are calculated using *numpy.linalg.eig* and from which X^{-1} is calculated.

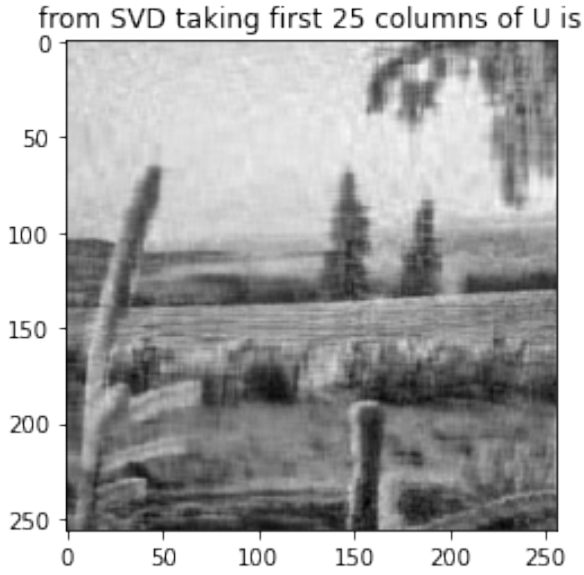
$$X^{-1} = \Lambda^{-1} X^{-1} A$$

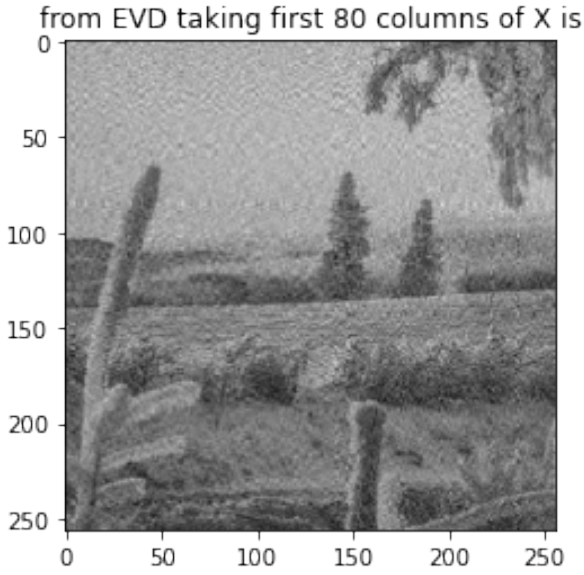
In the same way as above the image is reconstructed using K modes. But here since eigen values can be complex, for some values of K we will get complex image reconstruction when we don't take eigen values in conjugate pairs. We have taken care of that by checking imaginary part of eigen value and discarded those K values.

Experimental Results

We got all the singular values to be positive as expected.

We have taken 256*256 image and while reconstructing the image, we got almost original image with $k = 25$ using SVD values and to get the same clarity using EVD, we have to take nearly 80 modes.

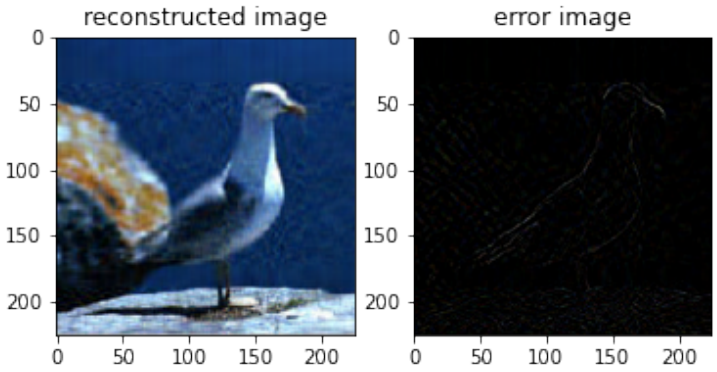




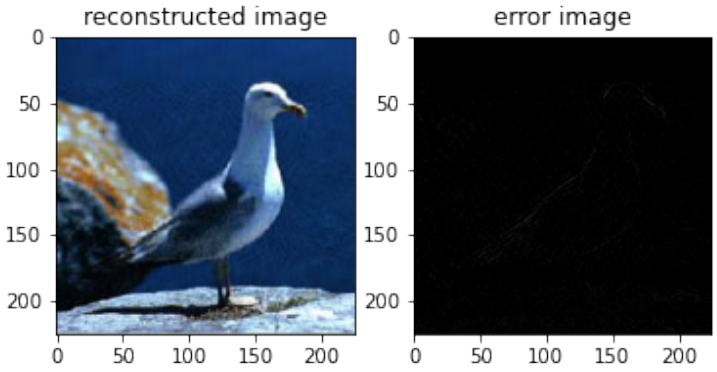
Performing the same on RGB images, I got similar value of modes, for which $k = 25$ gives almost original image from SVD. For RGB images we have to do

individual compression of R,G and B components and combine them back to get reconstructed image.

reconstructed image from SVD taking first 25 columns of U is

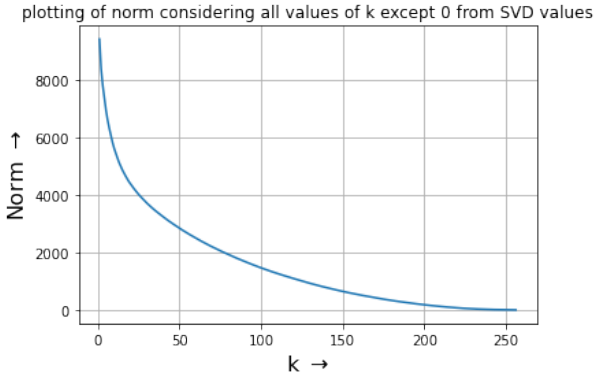


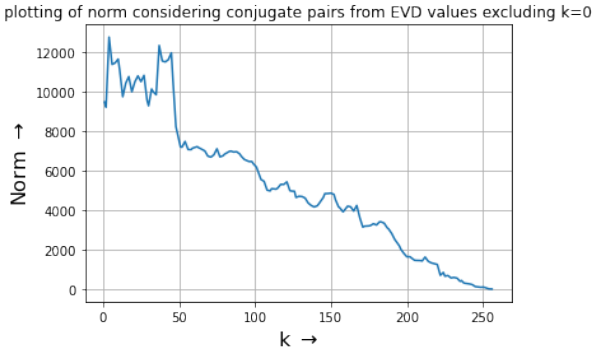
reconstructed image from SVD taking first 49 columns of U is

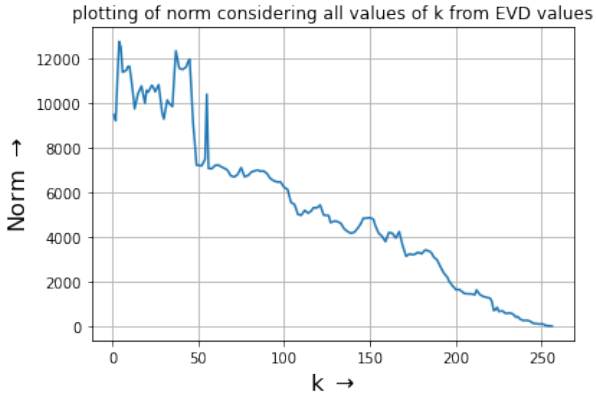


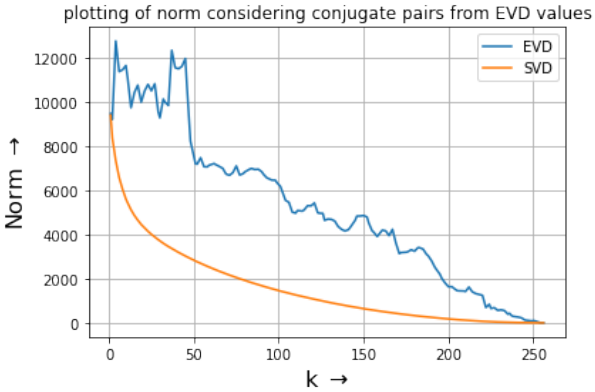
For further detailed images click here

we calculated the frobenius norms using *np.linalg.norm* for error image which is (actual image - approx image). For EVD values we plotted this both with taking and without taking care of conjugate pairs. Due to imaginary values in reconstructed image if we don't consider eigen values in pairs, we can see some peaks in norm.









For more graphs click here

Inferences

We successfully observed two compression methods using SVD and EVD.

We can use EVD for only square matrices as it involves taking inverse for X but SVD can be applicable even to rectangular matrices.

As we can see we can represent alomost exact image using SVD just by taking 25 singular values, so we just want 2*(256*25) + 25*25 which is 13,425 values to store the image.

But if we see using EVD values we should consider atleast 45 modes to get features of image and for a proper image we have to take above 80 modes. Which is atleast 2*(256*80) + 80*80 i.e. 47,360 values to store the image.

If we observe the graphs of norms it is clear that SVD norms are reducing smoothly and at faster rate compared to EVD values.

Eventhough both methods takes lesser values to store than original image which takes 256*256 i.e. 65,536 values, we are getting a better reduction with SVD (80 percent) than EVD (25 percent) making SVD compression technique best suitable.