# PRML Assignment 1

KSV Vineeth, CS19B080

## I. Motive

One of the advantages of SVD and EVD is they help in obtaining an approximate form of matrix using less number of bytes. The motive of these experiments is to see SVD, EVD in action, mainly on how well they can compress gray scale images. Also to notice the similarities and differences between EVD and SVD.

## IIA. SVD

Suppose we have a matrix $A_{mxn}$, it can be factorized as $U_{mxm}\Sigma_{mxn}V^*_{nxn}$. Where $U, V$ are unitary ($U^{-1} = U^*$) and $\Sigma$ is a diagonal matrix, in which the information about $A$ is arranged in descending order,i.e., most important information at top. So that we can recreate the approximate $A$ using top $k$ entries from $\Sigma$, first $k$ columns of $U$ and first $k$ rows of $V^*$, if k is resonably small then this data will take less space($= m*k+k*k+k*n$) than $mn$, this way it helps in compression. $U$ is the eigen vector matrix of $AA^*$, $V$ is the eigen vector matrix of $A^*A$, and $\Sigma\Sigma^*$ contains their eigen values. $AA^*$ is a hermitian matrix so it will have orthnomal eigen vectors, also it is a positive matrix hence non negative eigen values. Now coming to finding $U, V, \Sigma$ in Python: Using numpy.linalg.eigh we can get eigen values and vectors of $AA^*$, from this we can get $U$ and $\Sigma$, using them we can compute $V^*$ as $\Sigma^{-1}@U^{-1}@A$.

## IIB. EVD

Suppose we have a square matrix $A_{mxm}$ with m linearly independent eigen vectors $E_{mxm}$, it can be factorised as, $E_{mxm}\Lambda_{mxm}E^{-1}_{mxm}$. Where $\Lambda_{mxm}$ is a diagonal matrix, in which the eigen values of $A$ are arranged in descending order of magnitude. Similar to SVD we can recreate the approximate $A$ using top $k$ entries. If k is resonably small then this data will take less space than $m*m$, this way it helps in compression. We can use numpy.linalg.eig(A) to get eigen values and eigen vectors. Eigen values may be complex, but occurs in conjugte pairs, in such cases inorder to obtain real images we have to consider either both of the values or none.

## III. Experimental Results

### i) SVD Image compression:

From almost k = 20 onwards error image is barely visible to naked eye, but the reconstructed image is slighlty blurry. From around k = 40 reconstructed image is almost as clear as original image

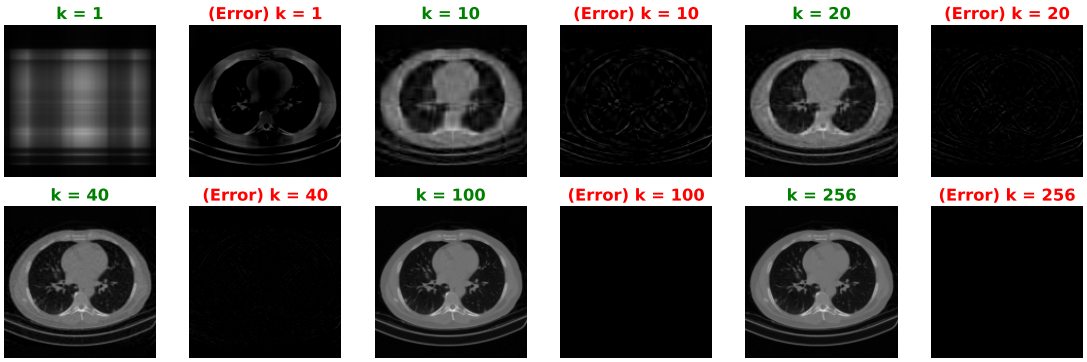


Figure 1: **SVD compressed images and their error counter parts**

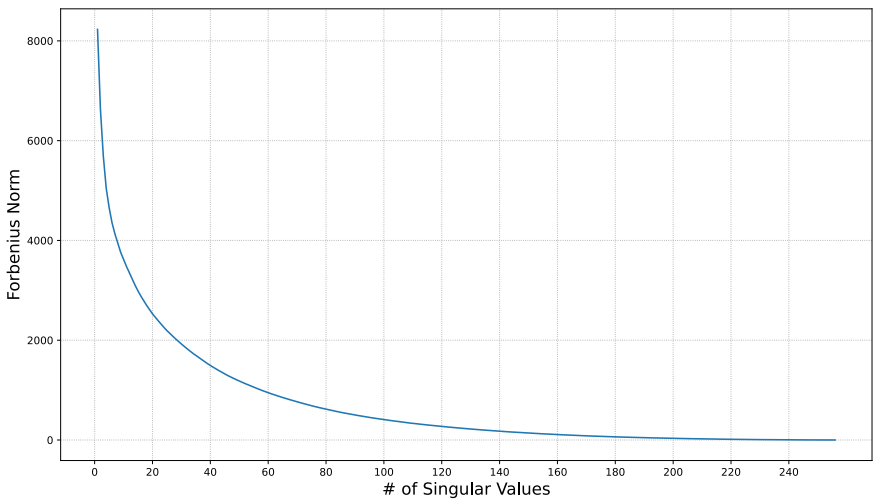


Figure 2: **Forbenius Norm vs. k**

## ii) EVD Image compression:

From almost k = 81 onwards error image is barely visible to naked eye, but the reconstructed image is slighlty blurry. From around k = 161 reconstructed image is almost as clear as original image

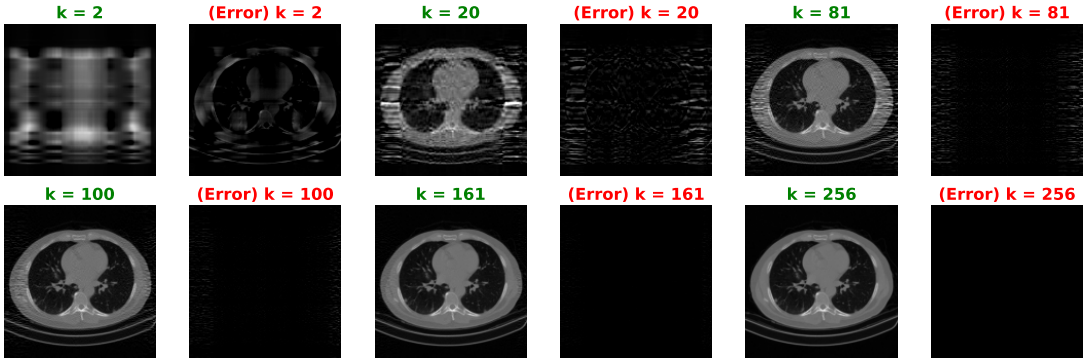


Figure 3: **EVD compressed images and their error counter parts**

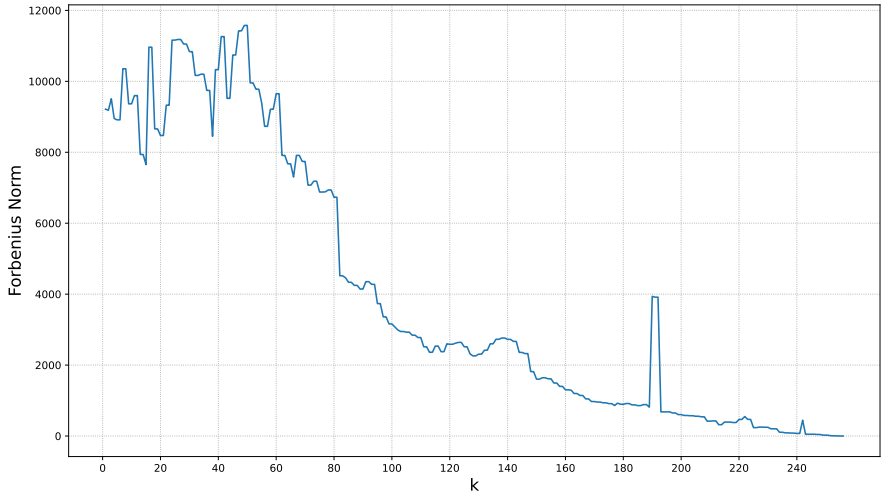


Figure 4: **Forbenius Norm vs. k**

## iii) SVD Ascending order of Singular values:

If we consider singular values in ascending order, we cannot see anything until we move very very close to 256. This shows that the larger singular values has a very important role. This can observed in the plot below.

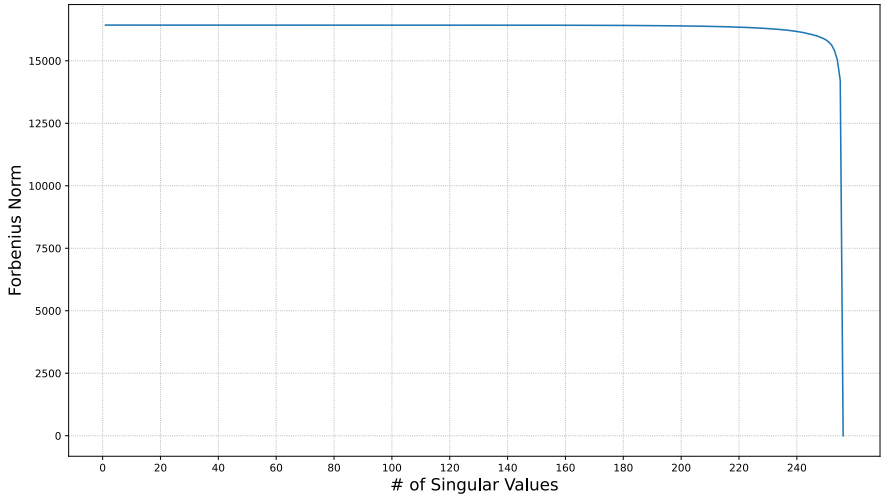


Figure 5: **Forbenius Norm vs. k**

## IV. Inference - SVD vs. EVD in the context of image compression

- Both are able to achieve image compression, although we needed way more number of Eigenvalues than Singular values to achieve the same quality. (Ex: to achieve < 3000 norm SVD needed around k = 20 where as EVD needed around k = 100). Hence SVD was able to compress the images to a smaller size than EVD
- The Forbenius Norm vs. K graph is buttery smooth for SVD, whereas there are lot of variations in the graph for EVD. On a higher level, both look concave upward and decreasing. Reason: In SVD $(U, V^T)$ are orthogonal, we are dividing the image into 256 Rank1 Matrices, they will also be orthogonal, this is the reason why as we adding more and more Rank1 matrices the error is continuously decreasing. This is no the case with EVD which is the reason for those variations.
- We can apply EVD only to square matrices having linearly independent Eigen vectors, whereas we can apply SVD to even rectangular matrices. This way SVD is more general than EVD.
- Both of them are decresing at a very rapid rate which shows that most of the data is stored in the first few Singular Values / Eigen Values. Comparitively SVD is decreasing way more rapidly than EVD.