1. PROJECT DETAILS:

IMAGE COMPRESSION OF JPG/JPEG/PNG/BMP FORMAT USING SINGULAR VALUE DECOMPOSITION(SVD)	
GROUP 21	
GROUP MEMBERS NAME	**ROLL NUMBER**
SHUBHAM PATEL	AU1940155
VRAJ KAVATHIYA	AU1940129
SHREY PATEL	AU1940110
VEDANT THAKORE	AU1940122

2. INTRODUCTION:

a) **Background –** The aim of our project is that to reduce/compress the size of the required image and create another image of reduced size and lower quality.

-The ultimate goal is that the size of the image should be reduced by making use of its pixels.

- Few modern used techniques for lossy image compression include:

1. Transform coding – This is the most commonly used method. Discrete Cosine Transform (DCT).
2. Chroma subsampling- is the practice of encoding images by implementing less resolution for chroma information than for luma information, taking advantage of the human visual system's lower acuity for colour differences than for luminance.
3. Fractal compression.
4. Using SVD (Singular Values Decomposition)

b) **Motivation and Overview –** In this project we have been using the concept of Standard value Decomposition i.e. SVD for lossy image compression. The name of the method used is **Power Method.**

- To give an overview the SVD approach is been done on each and every color of the image.
- Here the pixels of the image are represented in the form of m*n matrix and SVD is performed on it.
- The rank for approximation is to be set by user in the code as per image quality requirement. Higher the rank higher the resolution of compressed image.

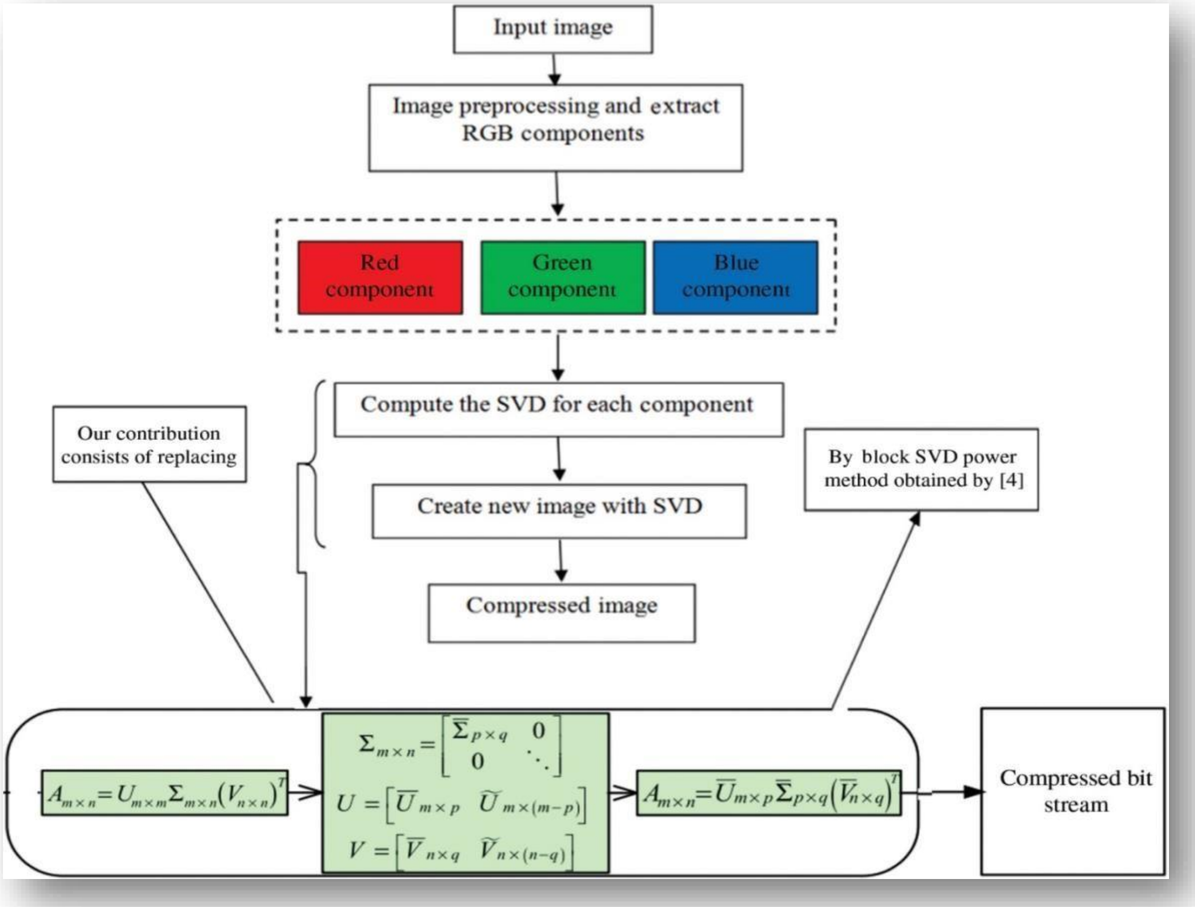
Pros	**Cons**
Can be used to compress any format of image e.g. jpeg, jpg, bmp etc. format	It is not a very good blockbased transformation technique, and using SVD on the whole image requires lots of memory.
Gives lossy compression	Takes lot of time. Time complexity is high
Gives option to choose rank to vary image quality.	To actually save data, we need k > (m*n / (m+n+1). That may not be the case with all the images
	Power method is not numerically stable as a small change in matrix could bring larger change in singular value ratio of two images.

3. APPROACH:

Basic Method:

- Power method is to be used for image compression.
- For compressing the coloured image the red, blue and green colour component need to be separated.

- The SVD is to be performed for red, blue and green colours. We can calculate how many bytes are needed to store the pieces of the rank approximation compared to size of original image.
- The program shows size of the original and compressed image in bytes, pixels of original image and the compression ratio(ratio of size of compressed to original image).


Input image
Image preprocessing and extract RGB components
Red component
Green component
Blue component
Compute the SVD for each component
Create new image with SVD
Compressed image
Our contribution consists of replacing
By block SVD power method obtained by [4]
$A_{m\times n}=U_{m\times m}\Sigma_{m\times n}(V_{n\times n})^T$
$\Sigma_{m\times n}=\begin{bmatrix}\overline{\Sigma}_{p\times q} & 0\\ 0 & \ddots\end{bmatrix}$
$U=\begin{bmatrix}\overline{U}_{m\times p} & \widetilde{U}_{m\times(m-p)}\end{bmatrix}$
$V=\begin{bmatrix}\overline{V}_{n\times q} & \widetilde{V}_{n\times(n-q)}\end{bmatrix}$
$A_{m\times n}=\overline{U}_{m\times p}\overline{\Sigma}_{p\times q}(\overline{V}_{n\times q})^T$
Compressed bit stream

Linear Algebra Concept: (Chen, Image Compression with SVD, Dec. 13, 2000)

-The basic concept is to represent an image with size ***m*** by ***n*** as a twodimensional ***m*** by ***n*** matrix.

-SVD is then applied to this matrix to obtain the **U**, **S**, and **V** matrices. **S** is a diagonal ***m*** by ***n*** matrix whose number of non-zero elements on the diagonal determine the rank of the original matrix.

-The fundamental concept of the SVD-based image compression scheme is to use a smaller number of rank to approximate the original matrix.

- If the original image is rectangular, then the matrix representation would require a ***m*n*** storage space.
- Applying SVD on the original matrix and use only ***k*** singular values results in a ***2r*k + k* (min of m and n)** storage space. In order to achieve the goal of compression, the ***k*** used for the re-constructed image would have to be smaller than ***m^2 / (1+2m).***
- Apart from these, we have created many functions like for dot product, transpose, inverse functions without using the inbuilt python libraries.

-

Then used these functions in the main SVD function for different colours and implemented it.

4. CODING AND SIMULATION:

Coding Strategy:

Algorithm of Power method:

We give a simple algorithm for computing the Singular Value Decomposition of a matrix A belongs to R^m*n. We start by computing the first singular value sigma1 and left and right singular vectors u1 and v1 of A

```
1. Generate random unit vector(x) having mean = 0 and standard deviation= 1

2. for i in [1,……,min(m,n)]:

3. xi ← (A-transpose)*Ax(i-1)

4. v1 ← xi / ||xi||

5. $\sigma_1$ ← ||A*v1||

6. u1 ← (A*v1)/ $\sigma_1$

7. return ($\sigma_1$; u1; v1)
```

KEY STEPS WHICH INCLUDE LINEAR ALGEBRA:

-Let A be the m x n matrix.

❼$A^T A = (U \sum V^T)^T (U \sum V^T) = V \sum U^T U \sum V^T = V \sum^2 V^T$

-We can eliminate U by performing A^TA. Let $B = A^T A = V \sum^2 V^T$ and x be the random unit vector.

-We can write it in terms of singular vectors $x = \sum_i c_i v_i$. Hence $Bx = \sum_i c_i \sigma_i^2 v_i$ and performing recursively k times it will give $B^k x = \sum_i c_i \sigma_i^{2\ k} v_{i.}$

-As the first singular value σ_1 is larger than all other singular values, the $B^k x$ corresponding to v_1 will be larger than rest others.

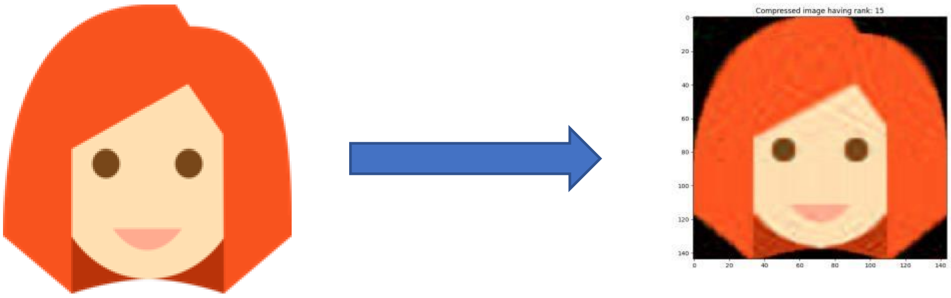
-Choosing random unit vector would ensure that for all vectors v_j, the ratio of σ_1 /σ_j would be greater than 1.

-The loop computes $x_{t+1} = Bx_t$, and renormalizing at each step. The loop will break when the angle between x_t and x_{t+1} is nearly 1. Using v_1, σ_1 and u_1 are calculated.

-

We want to ensure that the vectors in span of v_1 are ignored and hence we will set A to A = A – $\sigma_1 u_1 v_1^T$ and repeat the whole process again.

Figures:

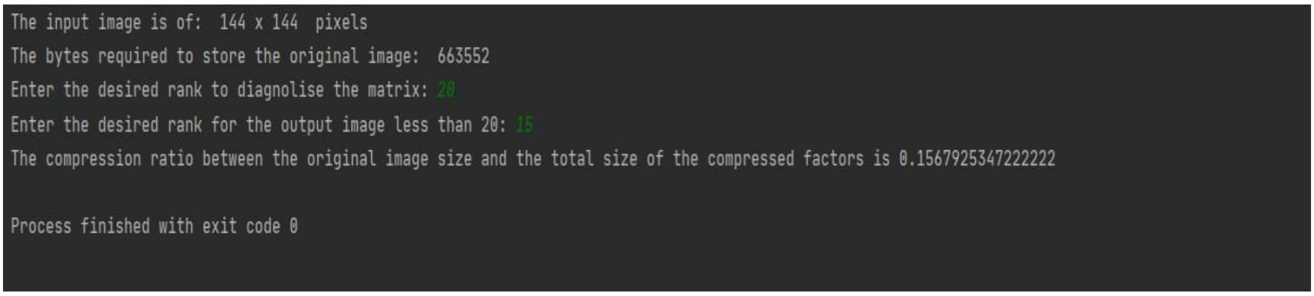


INPUT IMAGE SIZE – 81.1 KB

OUTPUT IMAGE SIZE – 63.9 KB

INPUT image is a .bmp image of size 81.1 kb and ouput image is of .bmp format of size 63.9 kb.

Rank 15 approximation is being used.



```
The input image is of:  144 x 144  pixels
The bytes required to store the original image:  663552
Enter the desired rank to diagnolise the matrix: 20
Enter the desired rank for the output image less than 20: 15
The compression ratio between the original image size and the total size of the compressed factors is 0.1567925347222222

Process finished with exit code 0
```

OUTPUT interface along with the compressed image.

Inferences:

-In our code the quality of the output image depends on value passed in svd function for red, blue and green color.

- Higher the value, higher the resolution.

- We ask for rank value (k) from user.

-More difference in image quality is observed by giving different k values at higher value passed in svd function of colors.

- But this requires higher amount of time, time complexity increases.

So we have to kept value passed in svd function 10, 15 or max 20.

-

- By doing so at different inputs of k, fluctuation in image quality is not seen.

-But if we pass 50 value in svd function, then a fair change is observed on changing k value.

5. CONCLUSIONS:

Closing remarks:

- Our group has finally worked on image compression with the help of power method which includes the concept of SVD(standard Value Decomposition).
- All types of image formats can be compressed for e.g. .bmp, .jpg , .jpeg etc.
- Apart from this there is a major drawback; that the program requires more time with increase in rank(k) value.
- The quality of the output image depends on the k value.
- **NOTE:** In case of .bmp images, the algorithm faster. Also higher reduction in size is observed for such image.

INDIVIDUAL CONTRIBUTION TO THE PROJECT	
NAME	**CONTRIBUTION**
Shrey Patel (AU1940110)	- Surfed on the internet for the approach and participated in the group discussion to decide the final method. Have gone through various articles and github. - In the project created functions of randomUnitVector and transpose and implemented in the svd function. - Made the abstract and the final report single handedly. - After the code was complete did the debugging and found the errors and helped Shubham and Vraj to resolve them.

Shubham Patel(AU1940155)	- Surfed on the internet for the approach and came up with the final approach of Power Method. - Designed the SVD function and implemented it for 3 different colors. - Also did the final implementation of SVD for compression. - Found out the errors after code was done and tried to resolve them.
Vraj Kavathiya(AU1940129)	-Gone through many websites and Github for approach. - After the code was written once, did the debugging and corrected the errors. - Created and implemented the dot, outer product and norm function. - Finalised ppt creation with Vedant.
Vedant Thakore(AU1940122)	-Along with Shrey created randomUnitVector and transpose function. - Designed the power point presentation for the final submission.

6. REFERENCES:

References

Chen, J. (2000, Dec 13). *Image Compression with SVD*. Retrieved from http://fourier.eng.hmc.edu/e161/lectures/svdcompression.html

Chen, J. (Dec. 13, 2000). *Image Compression with SVD.* ECS 289K Scientific Computation.

L. E. Liberty, “Lecture 7: Singular Value Decomposition,” *Yale.edu*. [Online]. Available: http://www.cs.yale.edu/homes/el327/datamining2013aFiles/07_singular_value_decomposition.pdf .

L. S. Arora, “Lecture 14: SVD, Power method, and Planted Graph problems (+ eigenvalues of random matrices),” *Princeton.edu*. [Online]. Available: https://www.cs.princeton.edu/courses/archive/fall14/cos521/lecnotes/lec14.pdf.

K. E. Asnaoui, “Image compression based on block SVD power method,” *J. Intell. Syst.*, vol. 29, no. 1, pp. 1345–1359, 2019.