Comparative Analysis of Image compression using SVD, DCT, 2D PCA and 2D block-block PCA algorithms

Manivas Kandukuri

M.Tech 1st year, Roll No:204102304

Abstract:

Images, used in Image Processing application, require big storage space and a considerable bandwidth for transmission. One of the possible solutions to this problem is data compression whose main goal is to reduce the quantity of data used to represent a digitized image. The data compression techniques always eliminate the redundant data and thus reduce the amount of space required to store an image. In this project, Singular Value Decomposition (SVD), Discrete Cosine Transform (DCT), 2D Principal Component Analysis (PCA) and 2D block-by-block PCA algorithms are used to compress an image and their performances are evaluated and compared. PSNR (Peak Signal to Noise Ratio) and Compression Ratio (CR) are the performance metrics used. In each algorithm, the image is compressed at different Compression ratios and corresponding PSNR values of the compressed image are calculated. For each algorithm, a plot of PSNR vs Compression Ratio is made to compare their performances.

Keywords: SVD;DCT;PCA;PSNR;Compression Ratio

1 Introduction

SVD is a linear matrix transformation used for compressing images. Using SVD an image matrix is represented as the product of three matrices U, S, and V where S is a diagonal matrix whose diagonal entries are singular values of matrix A. The image A can also be represented by using less number of singular values, thus, presenting necessary features of an image while compressing it. The compressed image requires less storage space as compared to the original image. To choose the value of k i.e. number of Eigen values for compression and reconstruction of the image is an important decision for acceptable reconstruction.

DCT is a technique for converting a signal into elementary frequency components. It is widely used in image compression. DCT expresses a sequence of finitely many data points in terms of a sum of cosine functions oscillating at different frequencies. DCT has a high energy compaction capability i.e.,

the most of the energy is concentrated in the low frequency components. This property of DCT is used in image compression. The high frequency components are suppressed and inverse DCT is applied on the suppressed image to get the compressed image

Principal Component Analysis (PCA) is a statistical approach that linearly transforms an original set of variables into a smaller set of uncorrelated variables that represent the largest variance from the multivariate input data. PCA has been widely applied in face recognition as it is used to extract the principal feature of the high dimensional data space. It has also received growing attention in image compression as its applications is supported by the idea that the 'important information' is captured by the principal components. In image compression, PCA is also known as Karhanen-Loéve (KL) Transform or Hotelling Transform where computation of variance-covariance matrix of the data is diagonalized using Singular Value Decomposition (SVD).

1.1 Background

A.Image compression through SVD:

By applying SVD on an image, the image matrix G is decomposed into 3 different matrices L, D and R.

$$G = LDR^T \tag{1}$$

where G is a m × n matrix and D is a m × n diagonal matrix in which the entries along the diagonal of D are singular values of G

$$G = l_1\sigma_1 r_1^T + l_2\sigma_2 r_2^T + ...l_r\sigma_r r_r^T ..... + l_N\sigma_N r_N^T \tag{2}$$

where r is the rank of G

For good amount of compression to be achieved, only the first k values of equation (2) are taken so that equation (2) becomes

$$G = l_1\sigma_1 r_1^T + l_2\sigma_2 r_2^T + ...l_k\sigma_k r_k^T \tag{3}$$

where k < r

B.Image compression through DCT:

Given that *f(m,n)* represent an image with the size NxN. *C(u,v)* be the coefficients obtained from the two-dimension DCT of the image, then:

$$C(u,v) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} f(m,n)\alpha(u)\alpha(v)\cos[(2m+1)u\pi/2N]\cos[(2m+1)v\pi/2N]$$

$$\alpha(u) = \sqrt{1/N}, u = 0 \text{ and } \sqrt{2/N}, u = 1, 2, ..., N-1$$
$$\alpha(v) = \sqrt{1/N}, v = 0 and \sqrt{2/N}, v = 1, 2, ..., N-1 \tag{4}$$

From equation (4), it can be seen that the workload of computing C(u,v) is increased as the image size is increased. So in actual application, we use the approach of image blocking to divide the image matrix with size of $N * N$ into $(N/h)^2 (h \leq N)$ image blocks $M_i (i = 1, 2, ..., (N/h)^2)$ with size of $h * h$.
DCT is performed on each M_i, which results in:

$$D_i = TM_iT' \tag{5}$$

where T is the transformation matrix derived from equation (4)

According to the requirement of image quality and compression rate,we can select the quantization matrix which is performed on equation (5). The quantized matrix contains many zero elements which represent no information. Accordingly, we can use a few non-zero elements to represent an image to realize the image compression.

Here there are many zero elements in the DCT coefficient matrix, and the coefficients $D_i(u, v)$ are small when coordinates of u and v are big. Moreover, the bigger coefficients are located in the left-top of DCT coefficient matrix where u and v are small. So the left-top corner is taken as the useful information area, and an area template A is used to pick up bigger coefficients by the dot multiply of A and $D_i(u, v)$.

$$H_i = A * D_i, \qquad i = 1, 2, ..., N$$

$$\text{and A} = \begin{bmatrix} 1 & 1 & 1 & 1 & .. & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & .. & 1 & 0 & 0 & 0 & 0 \\ : & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ : & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6}$$

For all values of i, inverse DCT is applied on the matrix H_i to obtain the compressed image.

C.Image compression through 2D PCA: The input image X is an M × N monochrome image whereby each element represents the intensity value. PCA is started by subtracting each element with the mean obtained along the row. It will be seen later that the mean along the row will be added back if original image is to be retrieved. Covariance is applied on the mean adjusted matrix, $\bar{X}$ to measure the linear relationships between each variable. The corresponding matrix is a square matrix (N ×N) and the eigen vectors and eigenvalues for

the matrix is calculated. It turns out that the eigen vector with the highest eigenvalue is the principal components (k) of the image. Once the number of principal components is determined, a feature matrix, V containing all the corresponding eigen vectors will be formed,

$$V = [\lambda_1, \lambda_2, ...\lambda_j]_{(Nxk)} \tag{7}$$

The transpose of the feature matrix will be multiplied with the transposed of the mean-adjusted matrix to obtain the compressed data with reduced dimensionality,

$$Y = [V^T * \bar{X}^T]_{(kxm)} \tag{8}$$

In image compression, it is of interest to know how reconstructed image might look with the compressed data. To reconstruct the image X with or without complete set of eigen vectors,

$$Y = V^T * (X - mean)^T \tag{9}$$

$$X = (V * Y)^T + mean \tag{10}$$

The degree of image degradation and compression rate depends on the number of principal components selected. The lesser the selected number of principal components, the more the data will be compressed at the expense of image quality.

D.Image compression through Block-by-block PCA: Instead of compressing the whole image at once, we are interested in working on the sub-block of the original image. Input images are partitioned into blocks of size n and PCA algorithm was applied individually on each block. Each block X_i consists of intensity values f (x, y) where i represent the block number of the image.

$$X_i = \begin{bmatrix} f(0,0) & f(0,1) & ... & f(0,n-1) \\ f(1,0) & f(1,1) & ... & f(1,n-1) \\ : & : & ... & : \\ f(n-1,0) & f(n-1,1) & ... & f(n-1,n-1) \end{bmatrix}$$

For each block, i, 2D PCA is applied on each block as per equation (7) and the block is reconstructed back using dominant eigen vectors using equation (10).

1.2 Terminologies

DCT = Discrete Cosine Transform
PCA = Principal Component Analysis
PSNR = Peak Signal to Noise Ratio
SVD = Singular Value Decomposition

1.3 Organization

In section-1, mathematical background of SVD, DCT, 2D PCA and 2D block-block PCA algorithms and their implementation details are provided. In the upcoming sections, we are going to discuss about the evaluation metrics used, algorithm formulation, the plots obtained and the final conclusions.

2 Datasets used

2.1 Dataset

All the above algorithms are applied on the JPG and PNG images shown in the next page and their performances are evaluated and compared.

2.2 Evaluation Metrics

A. Compression Ratio (CR):

CR is the ratio between size of original image (in bytes) to size of compressed image (in bytes) and is given below

$$CR = \frac{size(Imagefile)}{size(Compressedfile)}$$

B. Mean Square Error (MSE):

The MSE is the cumulative squared error between the compressed and original image

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Original_i - Compressed_i)^2$$

C. Peak Signal-to-Noise Ratio (PSNR):

PSNR is the quality measure of the compressed image. It can define using MSE as

$$PSNR = 20log_{10}(\frac{255}{\sqrt{MSE}})$$

[1] [2] [3].

Figure 1: sample JPG image1

Figure 2: sample JPG image2

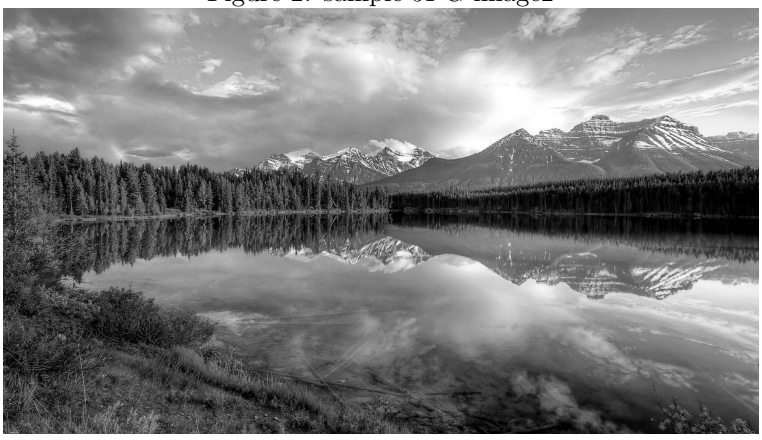

Figure 3: sample JPG image3

Figure 4: sample PNG image1

Figure 5: sample PNG image2

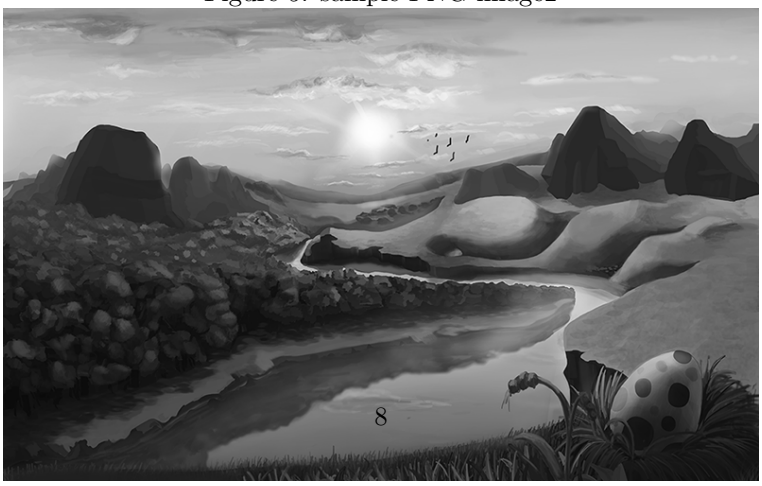

Figure 6: sample PNG image3

3 Methodology

3.1 Algorithm Formulation

A. SVD:

1. Apply SVD on the input image, and decompose it int into U,S and V^T matrices.
2. Take only the first K no. of columns in U, first K no. of singular values in S and first K no. of rows in V^T, form matrices U',S' and V^T'
3. The product of U',S' and $V^{T\prime}$ gives the compressed image.
4. Save the compressed image to a specified location in the disk.
5. Calculate the size of compressed image and size of original image using $os.stat().st_size$ command in python and then calculate the compression ratio.
6. Calculate the MSE between input image and compressed image and then compute PSNR from MSE.
7. Note the PSNR and Compression Ratio.
8. Repeat Steps No: 1 to 7 for different values of K.
9. A graph of PSNR vs Compression ratio is then plotted.

B. DCT:

1. The image is broken into nxn blocks of pixels. (In this project, DCT algorithm is implemented for block size (n) = 8 and 32).
2. Working from left to right, top to bottom, the DCT is applied to each block.
3. A quantization matrix of size nxn (with all 1's at the top left corner of the image and 0's elsewhere) is created using zig-zag coding.
4. Considering only K number of 1's in the quantization matrix, each DCT block is multiplied (element-wise) with the quantization matrix to eliminate the high frequency DCT coefficients.
5. Inverse DCT is applied on each quantized DCT block to get the compressed image.
6. Save the compressed image to a specified location in the disk.
7. Calculate the size of compressed image and size of original image using $os.stat().st_size$ command in python and then calculate the compression ratio.

8. Calculate the MSE between input image and compressed image and then compute PSNR from MSE.
9. Note the PSNR and Compression Ratio.
10. Repeat Steps No: 4 to 9 for different values of K.
11. A graph of PSNR vs Compression ratios (for block size, n=8 and n=32) is then plotted.

C. 2D PCA:

1. Considering the columns of the image as data vectors, calculate the mean vector.
2. Normalize the input image by subtracting the mean vector from each column of the image and form the covariance matrix using the normalized image.
3. Apply SVD on the covariance matrix and decompose it into U,S and U^T matrices.
4. Taking only the first K no. of columns in U and form the matrix V as per equation (7).
5. Obtain the compressed image as per the equations no. (8),(9) and (10).
6. Save the compressed image to a specified location in the disk.
7. Calculate the size of compressed image and size of original image using $os.stat().st_size$ command in python and then calculate the compression ratio.
8. Calculate the MSE between input image and compressed image and then compute PSNR from MSE.
9. Note the PSNR and Compression Ratio.
10. Repeat Steps No: 2 to 9 for different values of K.
11. A graph of PSNR vs Compression ratios is then plotted.

D. 2D block-block PCA:

1. The image is broken into nxn blocks of pixels. (In this project, 2D block-block PCA algorithm is implemented for block size (n) = 8 and 32).
2. Working from left to right, top to bottom, steps No: 1 to 5 of 2D PCA algorithm are applied to each block and compressed image is obtained.
3. Save the compressed image to a specified location in the disk.

4. Calculate the size of compressed image and size of original image using $os.stat().st_size$ command in python and then calculate the compression ratio.

5. Calculate the MSE between input image and compressed image and then compute PSNR from MSE.

6. Note the PSNR and Compression Ratio.

7. Repeat Steps No: 1 to 6 for different values of K.

8. A graph of PSNR vs Compression ratios (for block size, n=8 and n=32) is then plotted.

3.2 Plots tables and outputs

Following are the PSNR vs compression Ratio plots obtained for given set of JPG and PNG images using different image compression algorithms.

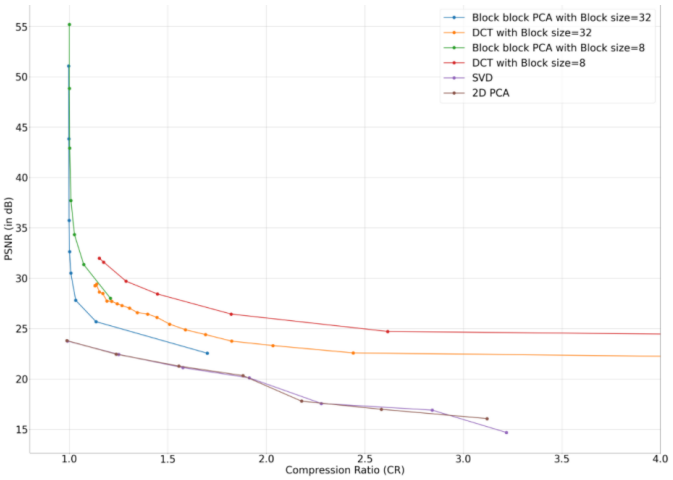

Figure 7: PSNR vs CR plot for sample JPG image1

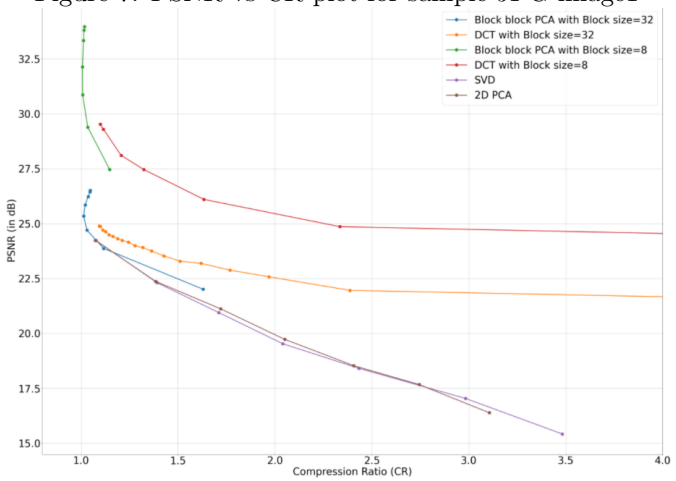

Figure 8: PSNR vs CR plot for sample JPG image2

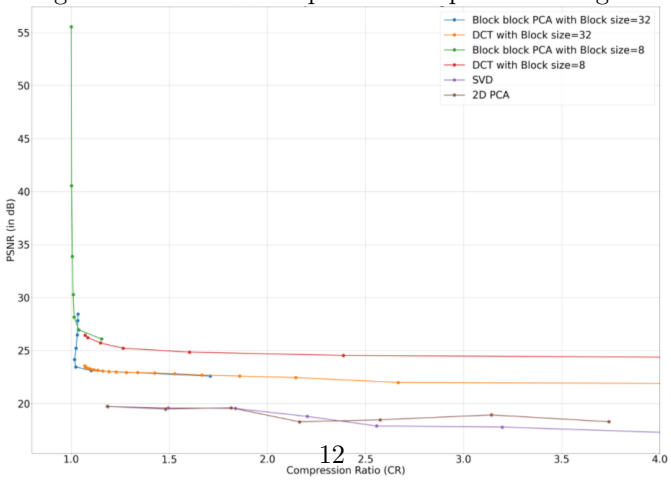

Figure 9: PSNR vs CR plot for sample JPG image3

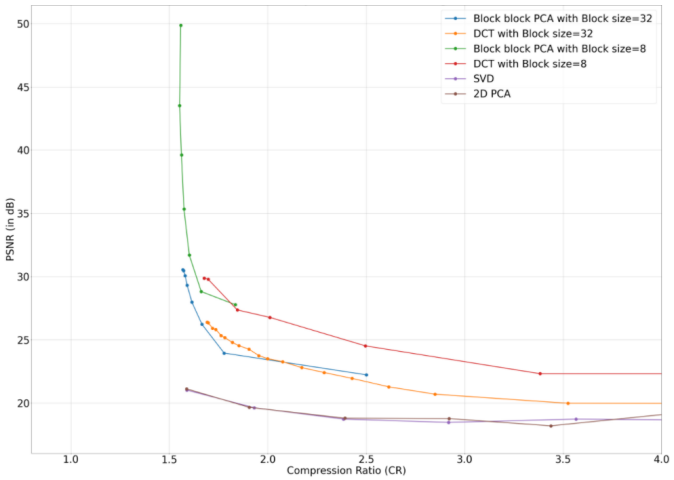

Figure 10: PSNR vs CR plot for sample PNG image1

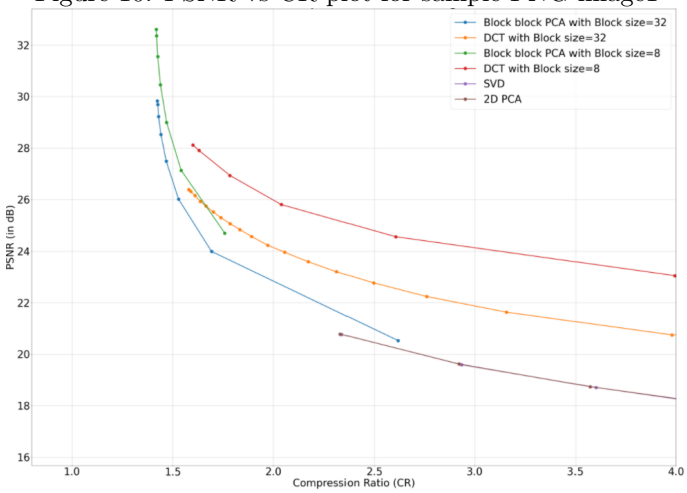

Figure 11: PSNR vs CR plot for sample PNG image2

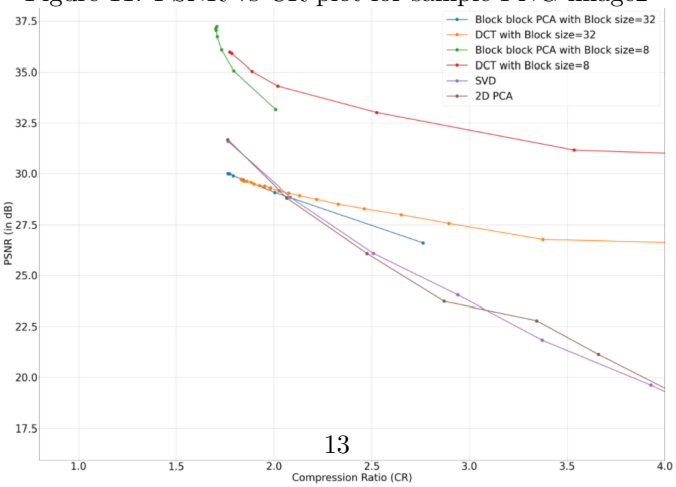

Figure 12: PSNR vs CR plot for sample PNG image3

4 Conclusions

From the plots of PSNR vs MSE, for the JPG and PNG images in the dataset,

1. Among all the techniques, quality of compression is high in case of DCT (with block size=8) followed by 2D block-block PCA (with block size=8).
2. In case of DCT and 2D block-block PCA algorithms, smaller block sizes give better compression quality compared to bigger block sizes.
3. The performance of 2D PCA algorithm is almost close to that of SVD algorithm and the performance of these two algorithms is lowest among all the algorithms.

References

[1] N. M. S.T. Lim, D.F.W. Yap, "Medical image compression using block-based pca algorithm," *International conference on Computer, Communication, and Control Technology*, 2014.

[2] H.-L. Z. BIN JIANG, GUO-SHENG YANG, "Comparative study of dimension reduction and recognition algorithms of dct and 2dpca," *Seventh International Conference on Machine Learning and Cybernetics, Kunming*, 2008.

[3] B. M. K. N. Prasantha H S, Shashidhara H L, "Image compression using svd," *International Conference on Computational Intelligence and Multimedia Applications*, 2007.