ICCSE'21 International Conference on Contemporary approach on revolutionary Science and Engineering, April 9-10, 2021 ISBN: 978-81-910765-1-6

A New Technique for Image Compression Using Linear Algebra with Python Algorithm

Dr.S.Karthigai Selvam[1] and Dr.S.Selvam[2]
[1]*Assistant professor, Department of Mathematics,*
N.M.S.S.V.N. College, Madurai – 625 019, Tamil Nadu, India
E-mail : s.karthic4@gmail.com
[2]*Head & Assistant Professor, Department of Computer Applications,*
N.M.S.S.V.N. College, Madurai – 625 019, Tamil Nadu, India
E-mail : s.selvammphil@gmail.com

Abstract: In recent days, the data are transformed in the form of multimedia data such as images, graphics, audio and video. A large amount of storage is needed for Image data. Minimize the redundancy of Image data using the compression concept such as Lossy Compression Techniques. Our proposed method shows the disadvantages of SVD function available in the Python language. Our proposed technique is compared to few other compression standard methods. This proposed method shows that the running time is very low. We proved that our proposed image compression methods is better than any other methods.

Keywords: Image Compression, Singular Value Decomposition, MSE, Lossy image compression, PSNR.

1 Introduction

The Singular Value Decomposition(SVD) is eigen-decomposition and helps to examine rectangular matrices. It used some applications such as data mining, search engines, digital image processing etc. The main aim of this paper is to show the SVD function used for image compression. Decreasing the storage helps to fast loading of images in the website and servers. It will reduce the loading time of the webpage. Our intention is to decrease the storage and present without lossy of data in the input image.

2 Existing Methods

Now-adays, there are various image compression techinques are used to storage images. An emprical study of few standard methods such as lossy and lossless techniques. Lossless compression gives, there is no data losses and also reserve the reconstructed images. We studied various work related to SVD with image compression methods.

Generally, SVD is a lossy compression technique which achieves compression by using a smaller rank to approximate the original matrix representing an image. Furthermore, lossy compression yields good compression ratio comparing with lossless compression while the lossless compression gives good quality of compressed images.

There are various works related to SVD with image compression methods. Awwal et al.[7] presented new compression technique using SVD and the Wavelet Difference Reduction. A technique based on Wavelet-SVD, which used a graph coloring technique in the quantization process, is presented in[8]. Ranade et al.[9] suggested a variation on SVD based image compression. This approach is a slight modification to the original SVD algorithm, which gives much better compression than the standard compression using SVD method.

3 Image Compression Technique Using SVD

The main idea is to produce similarity of image using less storage and compression using SVD function. Images are represented by matrices with every pixel in an image is called as element.

There are two algorithm are used namely Python SVD function and SVD Power Method.

3.1 Algorithm of Python SVD Function

We used to compute svd of a matrix in python. The svd function returns U,s,V .

- U has left singular vectors in the columns
- s is rank 1 numpy array with singular values
- V has right singular vectors in the rows -equivalent to V transpose in traditional linear algebra literature

The reconstructed of the original matrix.

```
reconst_matrix =
np.dot(U[:,:k],np.dot(np.diag(s[:k]),V[:k,:]))
def compress_svd(image,k):
"""
Perform svd decomposition and truncated (using k singular
                values/vectors) reconstruction
returns
--------
reconstructed matrix reconst_matrix, array of singular
values s
"""
U,s,V = svd(image,full_matrices=False)
reconst_matrix = np.dot(U[:,:k],np.diag(s[:k]),V[:k,:]))
return reconst_matrix,s
```

3.2 Algorithm of SVD Power Method

Input: *A matrix* $A \in (R)^{n\times m}$, *a block-vector* $V = V(0) \in R^{m\times s}$ *and a tolerance tol*

Output: *An orthogonal matrices*

$U = [u_1,u_2,...,u_s] \in R^{n\times s}$

$V = [v_1,v_2,...,v_s] \in R^{m\times s}$

and a positive diagonal matrix

$\Sigma = diag(\alpha_1,\alpha_2,...,\alpha_s)$

such that : $AV = U\Sigma$

While (err > tol) do

$AV = QR$*(factorization QR),*

$U \leftarrow Q(:, 1:s)$

(the s first vector colonne of Q)

 ISBN: 978-81-910765-1-6

ATU = QR,
V ← Q(:, 1 : s) and Σ ← R(1 : s, 1 : s)
err = ||AV - UΣ||
End

3.3 Proposed Image Compression Technique

The proposed work is to decreasing the duplication of the image and transfer data to effectively. Our proposed method is used to enhanced the Block SVD Power method and also developed an new algorithm for compression of image. Fig. 1 shows the new Architecture of Image Pre-Processing using SVD.

By using SVD, data can take first singular value. It has a huge amount of Image information. From this, some singular value is denoted image with lightly distinct from the original image. The original image may be color image with RGB color component (or) may be grayscale image.

In order to creating new image with Python SVD function as indicated in the Fig. 1, we use :

$$I_{comp} = U(:, 1:K) * \Sigma(1:K, 1:K) * (V(:, 1:K)^T) \quad (1)$$

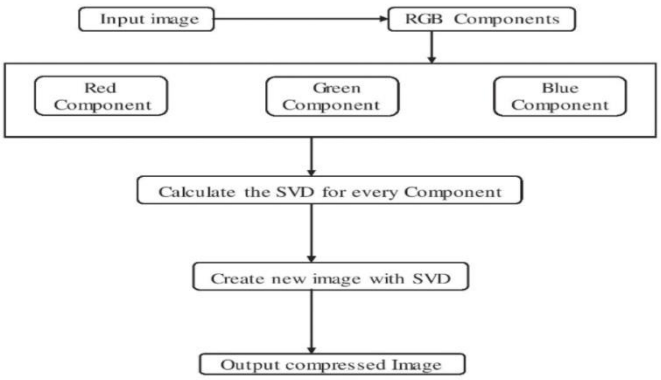

Fig. 1 New Architecture of Image pre-processing using SVD

Our new algorithm for Image compression that reduce few disadvantage of existing method of Python SVD function. Our proposed method modify the calculation of SVD for every component step and entries in the Image 1 are computing using Block SVD Power method by instead of Python SVD function and substitution of k rank determined by the equation (5).

The result proved that our proposed method is superior to any other compression techniques.

4 Experimental Results

Main aim of our proposed work is Image compression. Our experiments were performed on several images available on WANG Databases. Simulations were done in Python.

4.1 Measurement for comparison

To evaluate the performance of the proposed method, the quality of the image is estimated using several quality measurement variables like, Mean Square Error (MSE) and Peak Signal-to-Noise Ratio (PSNR). These variables are signal fidelity metrics and do not measure how viewers perceive visual quality of an image.

4.1.1 Measurement of Compression Ratio

The degree of data reduction obtained by a compression method can be evaluated using the compression ratio percentage (Q_{comp}) defined by the formula:

$$Q_{comp} = \frac{Size\ of\ Original\ image}{Size\ of\ Compressed\ image} \times 100 \quad (2)$$

4.1.2 Mean Square Error(MSE)

Percentage of MSE, which for two M ∗ N monochrome images X and Y where one of the images is considered noisy approximation of the other and is defined as follows:

$$e_{MSE} = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [X(i,j) - Y(i,j)]^2 \times 100 \quad (3)$$

4.1.3 Peak Singal-to-Noise Ratio (PSNR)

Percentage of PSNR is measured in decibels (dB), and is only meaningful for data encoded in terms of bits per sample bits per pixel. For example, an image with 8 bits per pixel contains integers from 0–255. PSNR is given by the following equation:

$$PSNR = 10\ log_{10} \frac{(2^B - 1)^2}{e_{MSE}} \times 100 \quad (4)$$

A high PSNR value indicates that there is less visual degradation in the compressed image.

4.2 Image Compression

We test our method, we develop a user interface. The method was applied to various and real images to demonstrate the performances of the proposed algorithm of image compression.

Here, we used 2 color images such as Giraffe and India Gate available in WANG Database and one in grayscale. Figures 4, 5, 6 and 7 show the test images and the resulting compressed images using Python SVD function [14] and the proposed compression method.

We recall that our goal is to approximate an image (matrix of $m \times n$) using the least amount of information. Thereby, to obtain a better quality of the compressed image using SVD, we use the *K* rank determined by El Asnaoui et al. [14]:

$$K = \frac{m \times n}{m + n + 1} \quad (5)$$

Where *m* and *n* are the size of original image.

 ISBN: 978-81-910765-1-6


(a) Girafe (b) India Gate

(c) grayscale

Fig. 4 Original images

4.2.1 Analysis with Color Image

After rank K = 438, we obtain:


(a) **(b)**

Fig. 5 Image compressed results obtained by: **a**. Python SVD function , **b**. Proposed method

Table 1 Image compression results for Giraffe.jpg, 1024 × 768, 858Kb, by using:

	Python SVD function			Proposed method		
K	Q_{comp}	MSE	PSNR	Q_{comp}	MSE	PSNR
25	0.0986	0.3082	0.3078	0.0750	0.4687	0.4800
50	0.0941	0.3122	0.3286	0.0741	0.4720	0.4968
75	0.0884	0.3346	0.3491	0.0739	0.4829	0.5076
100	0.0828	0.3550	0.3697	0.0737	0.4938	0.5185
125	0.0806	0.3696	0.3852	0.0736	0.5022	0.5266
150	0.0784	0.3841	0.4007	**0.0735**	**0.5107**	**0.5348**
175	0.0774	0.3960	0.4141	0.0735	0.5133	0.5373
200	0.0764	0.4079	0.4275	0.0735	0.5160	0.5399
225	0.0759	0.4187	0.4396	0.0734	0.5273	0.5505
250	0.0754	0.4296	0.4517	0.0734	0.5387	0.5612
275	0.0750	0.4402	0.4631	0.0733	0.5500	0.5724
300	0.0746	0.4508	0.4746	0.0733	0.5614	0.5836
325	0.0743	0.4614	0.4857	0.0733	0.5741	0.5982
350	0.0741	0.4720	0.4968	0.0733	0.5868	0.6128
375	0.0739	0.4829	0.5076	0.0732	0.6056	0.6380
400	0.0737	0.4938	0.5185	0.0732	0.6245	0.6633
425	0.0736	0.5025	0.5266	0.0732	0.6667	0.7162
438	**0.0735**	**0.5107**	**0.5348**	**0.0732**	**0.7089**	**0.7691**


(a) (b)

Fig. 6 Image compressed results obtained by: **a**. Python SVD function , **b**. Proposed method

Table 2 Image compression results for IndiaGate.jpg, 1024 × 768, 858Kb, by using:

	Python SVD function			Proposed method		
K	Q_{comp}	MSE	PSNR	Q_{comp}	MSE	PSNR
25	0.0954	0.2762	0.3390	0.0707	0.4372	0.4694
50	0.0923	0.2840	0.3510	0.0698	0.4451	0.4844
75	0.0860	0.2984	0.3629	0.0696	0.4616	0.4982
100	0.0797	0.3128	0.3749	0.0694	0.4782	0.5120
125	0.0773	0.3256	0.3854	0.0693	0.4921	0.5240
150	0.0749	0.3383	0.3960	**0.0692**	**0.5060**	**0.5359**
175	0.0737	0.3507	0.4062	0.0691	0.5108	0.5401
200	0.0725	0.3632	0.4165	0.0690	0.5155	0.5443
225	0.0717	0.3759	0.4270	0.0689	0.5374	0.5665
250	0.0709	0.3886	0.4375	0.0688	0.5593	0.5888
275	0.0706	0.4021	0.4487	0.0688	0.5802	0.6107
300	0.0702	0.4156	0.4599	0.0688	0.6011	0.6326
325	0.0700	0.4303	0.4721	0.0687	0.6227	0.6522
350	0.0698	0.4451	0.4844	0.0687	0.6442	0.6719
375	0.0695	0.4616	0.4982	0.0687	0.6760	0.7091
400	0.0693	0.4782	0.5120	0.0686	0.7078	0.7464
425	0.0693	0.4921	0.5240	0.0686	0.7776	0.8380
438	**0.0692**	**0.5060**	**0.5359**	**0.0687**	**0.8475**	**0.9297**

4.2.2 Analysis with Grayscale Image

In order to compare this performance, we also applied the new method to the gray scale image.
After rank K = 548, we obtain:


(a) **(b)**

Fig. 7 Image compressed results obtained on the: **a**. Python SVD function, **b**. Proposed method

Table 3 Image compression results for grayscale.jpg, 1024 × 960, 480Kb, by using:

	Python SVD function			Proposed method		
K	Q_{comp}	MSE	PSNR	Q_{comp}	MSE	PSNR
25	0.0498	0.8034	0.2767	0.0406	0.0953	0.3953
50	0.0497	0.7850	0.2922	0.0405	0.0945	0.3840
75	0.0464	0.5589	0.3108	0.0406	0.0751	0.3955
100	0.0431	0.3327	0.3293	0.0407	0.0558	0.4070
125	0.0420	0.2511	0.3440	0.0410	0.0452	0.4173

 ISBN: 978-81-910765-1-6

150	0.0410	0.1695	0.3587	0.0412	0.0346	0.4276
175	0.0407	0.1320	0.3714	0.0412	0.0285	0.4371
200	0.0405	0.0945	0.3841	0.0412	0.0224	0.4467
225	0.0406	0.0751	0.3955	0.0410	0.0186	0.4556
250	0.0408	0.0558	0.4070	0.0409	0.0148	0.4646
275	0.0410	0.0452	0.4173	0.0407	0.0124	0.4732
300	0.0412	0.0347	0.4276	0.0406	0.0100	0.4818
325	0.0412	0.0285	0.4371	0.0405	0.0084	0.4902
350	0.0412	0.0224	0.4467	0.0404	0.0068	0.4986
375	0.0410	0.0186	0.4556	0.0403	0.0057	0.5066
400	0.0409	0.0148	0.4646	**0.0402**	**0.0046**	**0.5146**
425	0.0407	0.0124	0.4732	0.0401	0.0038	0.5240
450	0.0406	0.0100	0.4818	0.0401	0.0030	0.5335
475	0.0405	0.0084	0.4902	0.0401	0.0024	0.5448
500	0.0404	0.0068	0.4986	0.0401	0.0018	0.5562
525	0.0403	0.0057	0.5066	0.0401	0.0013	0.5722
548	**0.0402**	**0.0046**	**0.5146**	**0.0401**	**0.0008**	**0.5883**

4.2.3 Analysis **with Other Methods**

To evaluate the robustness of our scheme, we test it with other methods like: [10, 13, 14]. Added experiment results for two images are listed in Table 4.

Table 4 Image comparison with various algorithms

	Color image (Fig. 4a)			Grayscale image (Fig.4c)		
	Q_{comp}	MSE	PSNR	Q_{comp}	MSE	PSNR
BTC method [13]	0.0927	0.6209	0.3020	0.0539	0.1611	0.2609
BTC method [10]	0.0734	0.0796	0.3912	0.0398	0.1902	0.3536
BTC method [14]	0.0672	0.0274	0.4375	0.0284	0.0347	0.4276
SVD method [14]	0.0735	0.0029	0.5348	0.0402	0.0047	0.5146
Proposed method	0.0731	0.0000	**0.7691**	0.0401	0.0008	**0.5883**

In this paper, the proposed algorithm is compared with the Python SVD function [14] and the other state-of-the-art algorithms.

Figs. 5, 6 and 7 shows that performance of our proposed method by using this two approach are same to original image. But quality of image is lacking in human visual performance is determined by the measurement such as PSNR, MSE and K rank.

In this experiment , calculated K rank for different images and also computed PSNR and MSE are tabulated in three tables namely Table 1, 2 and 3. It proved that the proposed method gives better performance compare to SVD. Table 4 proposed method is compared to other methods. The proposed method show that result is 1/3 of K rank compare to various method. It is able to produce a compressed image with good visual quality.

5 Conclusion

Our proposed method is used to overcome the limitations of existing methods used in the Python SVD process. The results showed that the proposed approach could be considered as a solution for the development of image abstraction. Our proposed method for image compression has been rapidly demonstrated due to the minimal number of repetitions in the compression algorithm.

Future Scope

The future purpose of this work is to use the SVD for statistical applications to detect the relationship between data, and to implement an abstract technique using the neural network in the area representing the clinical picture with the different threshold techniques associated with these multiwavelets.

References

[1] Madhuri A.J.: "Digital Image Processing. An Algorithmic Approach", pp. 175–217. PHI, New Delhi, 2006.

[2] Weinberger, M.J., Seroussi, G., Sapiro, G.: "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS", IEEE Transactions Image Processing **2**, pp. 1309–1324, 2000.

[3] Alkhalayleh, M.A., Otair, A.M.: "A new lossless method of image compression by decomposing the tree of Huffman technique", International journal of imaging & robotics **15**(2), pp. 79–96, 2015.

[4] Jianji, W., Nanning, Z., Yuehu, L., Gang, Z.: "Parameter analysis of fractal image compression and its applications in image sharpening and smoothing", Signal Processing: Image Communication journal pp. 681– 687, 2013.

[5] Bilgin, A., Michael, W., Marcellin, M., Altbach, I.: "Compression of electrocardiogram signal using JPEG2000", IEEE Transactions on Communications Electronics (ICIP) **49**(4), pp. 833–840, 2003.

[6] Awwal, M.R., Anbarjafari, G., Demirel, H.: "Lossy image compression using singular value decomposition and wavelet difference reduction", Digital Signal Process **24**, pp. 117–123, 2014.

[7] Adiwijaya, M., Dewi, B.K., Yulianto, F.A., Purnama, B.: "Digital image compression using graph coloring quantization based on wavelet SVD", Journal of Physics Conference Series **423**(1)", pp. 012-019, 2013.

[8] Ranade, A., Mahabalarao, S.S., Kale, S.: "A variation on SVD based image compression", Image and Vision Computing journal **25**(6), pp. 771–777, 2007.

[9] Doaa, M., Chadi, A.F.: "Image compression using block truncation coding". Cyber J. Multidiscip. J. Sci. Technol. J. Sel. Areas Telecommun. (JSAT), February, 2011.

[10] Delp, E.J., Mitchell, O.R.: "Image compression using block compression", IEEE Transactions on Communications **27**(9), pp. 1335–1342, 1979.

[11] Tsou, C.C., Wu, S.H., Hu, Y.C.: "Fast pixel grouping technique for block truncation coding", In: Workshop on Consumer Electronics and Signal Processing (WCEsp05), Yunlin, pp. 17–18 Nov, 2005.

 ISBN: 978-81-910765-1-6

[12] El Abbadi, N.K., Al Rammahi, A., Redha, D.S., Abdul-Hameed, M.: "Image compression based on SVD and MPQ-BTC", Journal 0f Computer Science **10**(10), pp. 2095–2104, 2014.

[13] El Asnaoui, K., Chawki, Y.: "Two new methods for image compression", International journal of imaging & robotics **15**(4), pp. 1–11, 2015.

[14] Bentbib, A.H., Kanber, A.: "Block power method for SVD decomposition", Analele Stiintifice ale Universitatii Ovidius Constanta Seria Matematica **23**(2), pp. 45–58, 2015.