International Journal of Engineering and Technology Innovation, vol. 11, no. 2, 2021, pp. 122-134

Image Compression Using Permanent Neural Networks for Predicting Compact Discrete Cosine Transform Coefficients

Saleh Alshehri*

Department of Computer Science and Engineering, Jubail University College, Saudi Arabia

Received 26 December 2020; received in revised form 20 February 2021; accepted 22 February 2021

DOI: https://doi.org/10.46604/ijeti.2021.6925

Abstract

This study proposes a new image compression technique that produces a high compression ratio yet consumes low execution times. Since many of the current image compression algorithms consume high execution times, this technique speeds up the execution time of image compression. The technique is based on permanent neural networks to predict the discrete cosine transform partial coefficients. This can eliminate the need to generate the discrete cosine transformation every time an image is compressed. A compression ratio of 94% is achieved while the average decompressed image peak signal to noise ratio and structure similarity image measure are 22.25 and 0.65 respectively. The compression time can be neglected when compared to other reported techniques because the only needed process in the compression stage is to use the generated neural network model to predict the few discrete cosine transform coefficients.

Keywords: image compression, discrete cosine transform, neural networks

1. Introduction

Image compression is an important field, especially in social media applications. In these applications, there are many requirements. Since most social media applications run on smartphones and personal devices, media content size and device processing speed are the most important requirements. These two requirements relate to each other, where a change in one affects the other. For example, if the media content is large, the hardware processing time will increase as compared to small-sized media content. Therefore, media content size should be as small as possible so that it can be stored and transmitted with minimal storage and processing power requirements.

Despite advances in information technology and, in particular, in the storage and processor industries, digital visual and audio systems are always speed- and storage-hungry. This is because there are continuous advances in smart phones and their applications, such that better image and video quality are improved alongside advancements in hardware designs. For example, in 2021, it is assumed that multimedia video content will need about 70% of communication bandwidth [1]. This is an important observation because most social media applications run on smartphones. While smartphones are always improving, they face processor speed limitations [2] due to the difficulty in processor cooling [2]. Mobile phone processors can be improved and made more powerful in terms of speed, but the heating problem can impose the usage of a thermal management system, where the processor speed has to be reduced [2]. Keeping in mind the consumer's need for better smartphone cameras and audio hardware, it is necessary to search for alternative and supporting solutions. Current mobile phone cameras have high resolution capability and can capture images of up to, and larger than, 4000×3000 pixels in size [3]. A color image of such size requires about 36×106 bytes of unsigned integers to be stored in raw pixels. Storing many such images requires large storage.

* Corresponding author. E-mail address: saaas101@gmail.com

Transmitting one such image also requires high transmission bandwidth and high processing capability. Therefore, it is always preferable to represent images in small sizes as much as possible. This leads to the study of image compression techniques.

Image compression is the process of reducing the image size while trying to keep the image appearance quality as of the original image. There are two main techniques: lossless, and lossy image compression. In the lossless image compression technique, the compressed and decompressed image pixels are exactly the same [4]. If compressed and decompressed images differ in the number of pixels, the technique is known as lossy image compression, in which some pixel information is lost [4]. Usually, lossy image compression methods produce a higher compression ratio (CR), which means the compressed images are of small size. However, lossless image compression methods are always preferable because they preserve pixel information. Many medical applications use lossless image compression methods for better diagnosis [5-6]. Many applications have been built on lossy image compression techniques, as long as the decompressed image appearance quality is acceptable.

Image transformation techniques—such as the fast Fourier transform (FFT), the discrete cosine transform (DCT), and the wavelet transform (WT)—are currently the most widely used techniques in the image compression field. DCT, for example, is integrated into many current image compression technique standards, such as JPEG [7-8]. The main disadvantage of such a technique is the processing time it takes. This can be noticed from Eq. (1), in which the basic technique requires the cosine to be calculated, and multiplications and double summations are to be performed. Recent methods use standard matrices to calculate DCT coefficients without performing the cosine calculation every time. However, many matrix multiplications are still required to calculate DCT coefficients and quantization, in addition to the integer coding step.

Lossy compression methods can be implemented using pattern recognition methods. Image pixels removed during compression can be recovered intelligently. A vast number of pattern recognition methods has been reported [9-12], including artificial neural networks (NN), Support Vector Machine (SVM), Self-Organizing Map (SOM), Principle Component Analysis (PCA), etc. Each of these has advantages and weaknesses.

The use of neural networks in image compression is based on training the NN on the image pixels, using fewer hidden nodes and output weights [9, 13-15]. These hidden nodes and output weights represent the compressed image. To decompress it, the hidden nodes are passed through the output weights to get the decompressed image. The size of the hidden nodes and the output weight represent the CR [16-19]. Image pixels can be removed and returned back, using the NN approach [20-21]. Another method is to train the NN on the DCT coefficients of the image, after it is decomposed into many blocks [16-19]. A block size of 8×8 pixels is not uncommon. While the NN performance in image compression is acceptable, the drawback is that training must be done for every image to be compressed, and that usually consumes time. Another shortcoming of NN is that a high CR may not be achieved in many cases [13, 19, 22].

In this research, both NN and DCT techniques are investigated to mitigate the shortcomings of each method and to utilize the advantages of both techniques. The proposed technique relies on building a general purpose and permanent NN that can be used to estimate the DCT coefficients so that neither training nor recalculating DCT coefficients are needed each time an image is to be compressed.

2. Predicting DCT Coefficients Using Neural Networks

DCT is being used extensively for image compression. Any multidimensional signal (an image, in this case) can be represented by many data points, as the sums of cosine functions oscillating at various frequencies [8]. These points are the DCT coefficients. The image can be reconstructed completely by calculating the inverse of DCT with number points less than the original image number of pixels. The image can still be reconstructed when using a reduced number of DCT coefficients. The quality of the reconstructed image is directly proportional to the number of DCT coefficients used. Usually, DCT is applied on blocks of an image with a preferable size of 8×8 pixels [7].

Thus, in order to compress an image, DCT coefficients must be calculated for each 8×8 block. After that, a few other steps must be performed, such as quantization and coding [7]. However, quantization step is used to make most of the high frequencies reduced to zero. The number of DCT coefficients needed to be processed is proportional to the chosen image quality. Choosing the percentage of coefficients is allowed in order to control the CR. However, a higher CR affects the decompressed image quality [7]. The process should be done for every image to be compressed involving DCT. In order to simplify the proposed technique, the quantization step is not implemented in the proposed method, which prevents from further controlling the CR and decompressed image quality. However, quantization is currently under investigation and will be presented in an upcoming research article.

In this research, a reasonably general NN model is built, using image block pixels as input feature vector samples and DCT coefficients as targets. This produced NN model is used each time an image is to be compressed, without recalculating the DCT coefficients. The process starts by collecting many images. More than 11,000 color images with various sizes are used [23]. These images are converted to grayscale images. The proposed technique is applied to color images in a later publication. Each image is then divided into 8×8 non-overlapping blocks. DCT is calculated for each block producing 8×8 coefficients. Eq. (1) shows the DCT calculation for a block of size 8×8. Out of these coefficients, the first 2×2 {DCT coefficients (1,1), (1,2), (2,1), (2,2)} are retained. Eq. (2) is used to reconstruct the block pixels values.

$$DCT(k,l)=\sum_{n=0}^{7}\sum_{m=0}^{7} im(n,m)\cos[\pi(2n+1)\frac{k}{16}]\cos[\pi(2m+1)\frac{l}{16}] \tag{1}$$

$$Decomp_{im(n,m)}=\frac{1}{64}\sum_{n=0}^{7}\sum_{m=0}^{7} DCT'(k,l)\cos[\pi(2n+1)\frac{k}{16}]\cos[\pi(2m+1)\frac{l}{16}] \tag{2}$$

where $0 \le k \le 7$; $0 \le l \le 7$ $im(n,m)$ is the image pixels values

$$DCT'(k,l)=\begin{cases}\frac{DCT(0,0)}{4} & k=0,l=0\\ \frac{DCT(k,0)}{4} & k>0,l=0\\ \frac{DCT(0,l)}{4} & k=0,l>0\\ \frac{DCT(k,l)}{4} & k>0,l>0\end{cases} \tag{3}$$

Since in this proposed method only four coefficients are needed, few calculations are performed. They are as follows:

$$DCT(1,1)=\sum_{n=0}^{7}\sum_{m=0}^{7} im(n,m)\cos[\pi(2n+1)\frac{1}{16}]\cos[\pi(2m+1)\frac{1}{16}] \tag{4}$$

$$DCT(1,2)=\sum_{n=0}^{7}\sum_{m=0}^{7} im(n,m)\cos[\pi(2n+1)\frac{1}{16}]\cos[\pi(2m+1)\frac{2}{16}] \tag{5}$$

$$DCT(2,1)=\sum_{n=0}^{7}\sum_{m=0}^{7} im(n,m)\cos[\pi(2n+1)\frac{2}{16}]\cos[\pi(2m+1)\frac{1}{16}] \tag{6}$$

$$DCT(2,2)=\sum_{n=0}^{7}\sum_{m=0}^{7} im(n,m)\cos[\pi(2n+1)\frac{2}{16}]\cos[\pi(2m+1)\frac{2}{16}] \tag{7}$$

below is an example of an 8×8 block of one image of the dataset and its corresponding DCT coefficients.

$$Block_i = \begin{pmatrix} 92 & 97 & 101 & 103 & 103 & 104 & 105 & 106 \\ 92 & 97 & 101 & 102 & 103 & 104 & 105 & 106 \\ 94 & 98 & 100 & 101 & 103 & 104 & 106 & 106 \\ 94 & 97 & 100 & 100 & 102 & 103 & 105 & 106 \\ 94 & 96 & 97 & 97 & 98 & 100 & 102 & 103 \\ 93 & 94 & 95 & 96 & 97 & 97 & 99 & 101 \\ 91 & 94 & 95 & 97 & 97 & 97 & 98 & 100 \end{pmatrix} \tag{8}$$

$$DCT\ of\ Block_i = \begin{pmatrix} 792.00 & -24.67 & -5.67 & -6.14 & -1.75 & -1.44 & -0.627 & 0.23 \\ 17.89 & -5.47 & -1.64 & -1.38 & -2.05 & 0.26 & -0.02 & 0.22 \\ -2.46 & 0.04 & -4.8 & -1.42 & 0.75 & -0.36 & 0.87 & -0.83 \\ -4.04 & 0.97 & 2.48 & -0.54 & 0.72 & 0.37 & 0.08 & -0.51 \\ 1.25 & 0.03 & -1.12 & 0.182 & -0.50 & 0.27 & 0.48 & 0.13 \\ 1.19 & 0.30 & 0.01 & -0.23 & 0.12 & 0.12 & -0.44 & -0.08 \\ -0.06 & 0.33 & 0.18 & 0.71 & 0.50 & 0.07 & -0.20 & 0.27 \\ 0.01 & 0.11 & 1.02 & 0.20 & -0.36 & -0.07 & -0.38 & -0.61 \end{pmatrix} \tag{9}$$

The only needed coefficients are:

$$DCT_{coff} = \begin{pmatrix} 792.000 & -24.670 \\ 17.892 & -5.473 \end{pmatrix} \tag{10}$$

The total number of samples becomes more than 6×106 samples. It is enough to randomly select only 350,000 samples to train the NN using the backpropagation model in order to predict the DCT 2×2 coefficients. Once the NN is fully trained, it is used to predict the values of these four DCT coefficients. These numbers represent the compressed image. To decompress it, the inverse DCT is applied on each of the four values of this collection of DCT coefficients, and the decompressed image is generated.

NN is a rigid yet simple model that can be used for classification and regression. The principle construction of the backpropagation model lies on updating the free parameters of the model based on the error generated from prediction. As shown in Fig. 1, each link in the NN algorithm between two layers of neurons represents a number called weight. In many applications, two layers are enough to construct an efficient NN model [24]. One set of weights connects the inputs to the hidden nodes, while the other set of weights connects the hidden nodes to the output node.

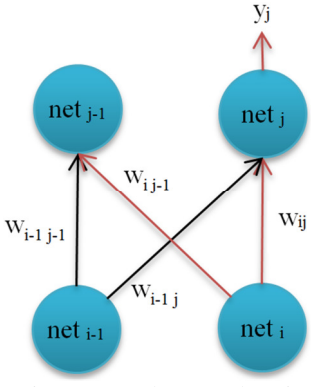


Fig. 1 Neural network unit

In NN, the node activation function is:

$$net_j = \sum_i y_i w_{ij}, y_j = f(net_j) \tag{11}$$

In Eq. (11), the transfer function $f(net_j)$ can be any smooth, differentiable, and nonlinear function. The logistic function $(1 + e^{-x})^{-1}$ is used in this study.

The errors are calculated as:

$$E = \frac{1}{2}\sum_i (t_j - y_j)^2 \tag{12}$$

where t_j is the target output and y_j is the predicted output. For the output layer, the errors are calculated as:

$$\frac{\partial E}{\partial y_j} = (y_j - t_j), \delta_j = \frac{\partial E}{\partial net_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial net_j}(y_j - t_j) f'(net_j) \tag{13}$$

The backpropagation error for node i in Eq. (13) is calculated as:

$$\frac{\partial E}{\partial y_j} = \sum_j \left(\frac{\partial E}{\partial net_j}\frac{\partial net_j}{\partial y_j}\right) = \sum_j (\delta_j w_{ij}), \delta_j = \frac{\partial E}{\partial net_i} = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial net_i} = \frac{\partial E}{\partial y_i} f'(net_i) \tag{14}$$

Finally, the weights are updated as:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}} = \delta_j y_j, \Delta w_{ij} = -\mu \frac{\partial E}{\partial w_{ij}} \tag{15}$$

where μ in Eq. (15) is the learning rate, which is less than 1.0.

Following Eq. (11) to Eq. (15) leads to y_j being very close to t_j within a minimum possible difference error. When reaching this stage, the NN is said to be either the regression or classification general predictor. In this research, the application needs the regression NN model. Fig. 2 shows the NN built for this purpose.

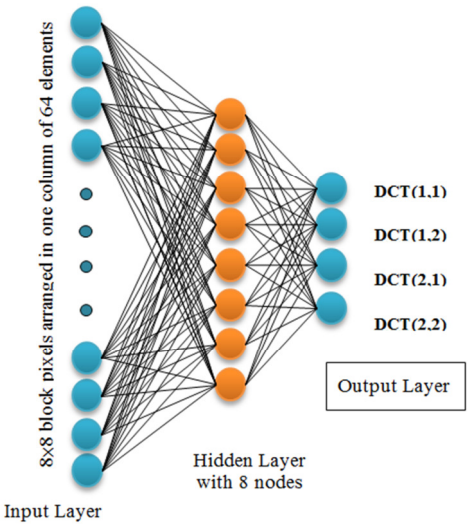
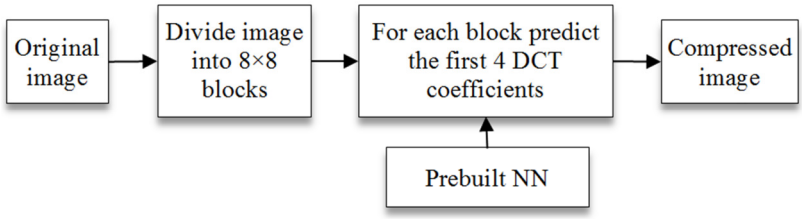


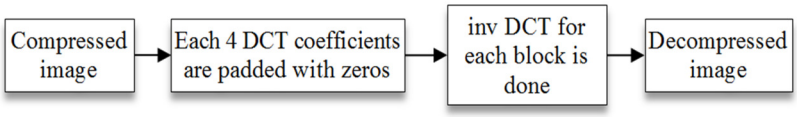
Fig. 2 The NN built to predict the DCT coefficients

Fig. 3 shows the proposed method of image compression and decompression stages. The NN is trained successfully. The size of the hidden layer cannot be determined precisely. Use of the NN growing and pruning algorithm is the common practice for determining the best hidden layer size [24]. Hidden layer sizes from 4 to 8 are investigated, and are found to produce almost

similar NN performance. Only sizes 3 and lower produced unsatisfactory results where DCT(2,1) and DCT(2,2) are not predicted by NN. The NN shows various performance accuracy for various hidden layer sizes. Fig. 4 shows the results for number of hidden nodes ranging from 4 to 10 nodes. The difference in performance accuracy between 8 nodes and 4 nodes is about 8%. For number of nodes exceeding 8, there is no much improvement. To demonstrate the proposed method, a generated NN model has 64×8 hidden free parameters in matrix form and 8×4 output free parameters in matrix form; this size gives the best results among all the others. These two matrices are the only needed information to generate the needed DCT 2×2 coefficients from any image that needs to be compressed. Generating these DCT coefficients involves only matrix multiplication, which can be performed easily and efficiently in any mobile phone or mobile device. In the receiving device, the received DCT coefficient must be used to reconstruct the decompressed image. Eq. (2), which computes the inverse DCT, is used for this purpose. It can be seen that using a hidden layer with 8 nodes does not outperform calculating the DCT coefficients directly because the number of arithmetic operations is higher. However, when using 4 hidden nodes, the number of arithmetic operations is comparable to that of the direct DCT coefficients calculation. It is worth mentioning that, if the DCT coefficients are increased to 3×3, then the number of direct DCT calculations is double the number of calculations based on NN predictions. This, in fact, promotes the use of NN in this application.



(a) Image compression stage



(b) Decompression stage

Fig. 3 The proposed method of image compression and decompression stages

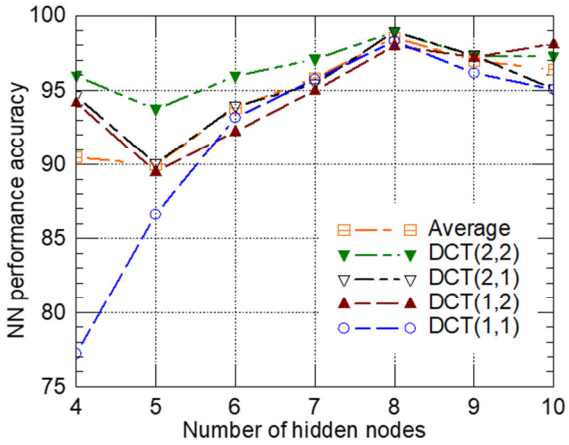


Fig. 4 Performance accuracy of NN for various hidden layer sizes

3. Results

To evaluate the compression quality, two metrics are used: peak signal to noise ratio (PSNR), and structure similarity image measure (SSIM), as seen in Eq. (16) and Eq. (18) [25].

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE}\right) \tag{16}$$

$$MSE = \frac{1}{RxC}\sum_{i=0}^{R-1}\sum_{j=0}^{C-1}(im(i,i) - Dim(i.i))^2 \tag{17}$$

where

MSE: is mean square error

im: is the original image

Dim: is the decompressed image

R and C: are image dimensions

$$SSIM = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{18}$$

where μ_x ad μ_y are averages of the two images, σ_x and σ_y are the variance of the two images, and c_1 and c_2 are two variables to stabilize the division with weak denominator.

While the CR is defined in many ways, Eq. (19) is used to calculate the CR as a percentage [12]. So, compression percentage and CR are used here interchangeably.

$$CR = 100(1 - \frac{compressed\ image\ size}{original\ image\ size}) \tag{19}$$

$$NN\ performance\ accuracy = 100\ (1 - \frac{\sum_{i=1}^{N}(prediction_i - target_i)}{N}) \tag{20}$$

where N is the data size.

When 2×2 DCT coefficients are built from an 8×8 block size, the CR is found to be 93.8% by dividing 4 (the number of DCT coefficients) by 64, which is the block size. The NN is evaluated randomly on 5,000 images from the dataset. The average performance accuracies are PSNR= 22.25 and SSIM=0.65.

To improve the compression quality, various block sizes are investigated. Specifically, block sizes of 3×3, 4×4, 5×5, 6×6, 7×7 and 8×8 are investigated. Each block size produces different CR, PSNR, and SSIM values. Table 1 and Fig. 5 show the result of using these various block sizes, and their performances based on Eq. (19) and Eq. (20). PSNR and SSIM decrease as CR increases by increasing the block size. The maximum obtainable CR is 94%. It can be obtained with PSNR less than 23. The table also shows the training and testing accuracy of the NN. Its compression quality is directly checked using the images to construct the DCT coefficients. It is found that PSNR and SSIM are identical to the PSNR and the SSIM generated using the proposed method, with differences on the order of 10^{-3}. This proves that NN can be used successfully to replace the calculation of DCT coefficients each time an image is to be compressed.

Table 1 Performance accuracy in NN training and prediction applied of images dataset

Block Size	CR %	NN Performance Accuracy				PSNR	SSIM
		DCT (1,1)	DCT (1,2)	DCT (2,1)	DCT (2,2)		
3 × 3	55	99.31	99.04	99.40	99.42	30.30	0.93
4 × 4	75	99.76	99.73	99.71	99.85	27.42	0.87
5 × 5	84	99.56	99.70	99.69	99.62	25.71	0.81
6 × 6	89	98.13	97.72	97.78	98.57	24.66	0.76
7 × 7	92	97.96	97.18	98.61	98.80	23.88	0.71
8 × 8	94	98.28	97.98	98.92	98.92	23.28	0.68

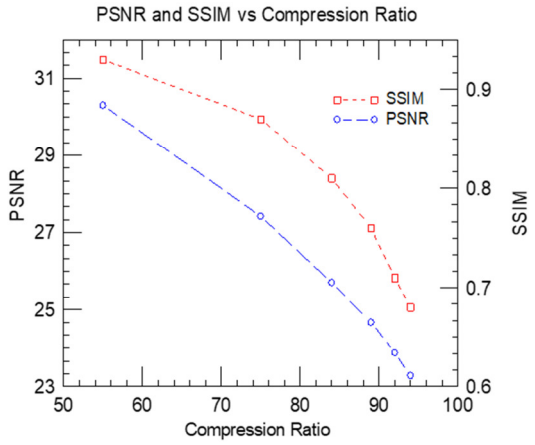
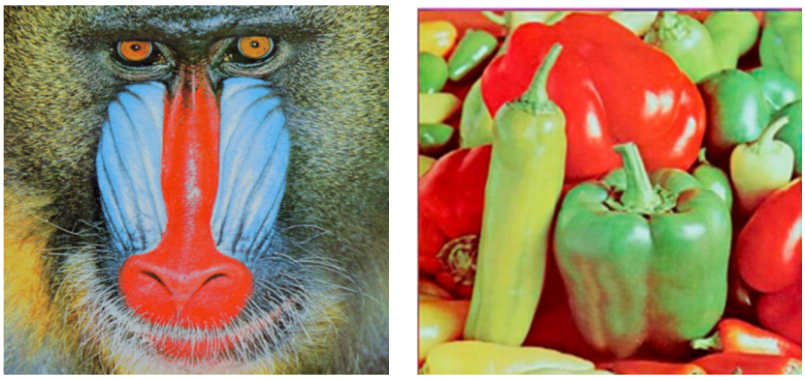


Fig. 5 CR vs compression performance on the image dataset

Fig. 6 shows well-known images, both of which are used to investigate the proposed method performance. Figs. 7(a)-(f) show the result of using the proposed method on the two images in Fig. 6. Figs. 7(a) and 7(b) are the original images; Fig. 7(c) is the Baboon image with a 3×3 block size which produced PSNR = 28.17 and SSIM = 0.88, while Fig. 7(d) is the Pepper image with a 3×3 block size which produced PSNR = 44.25 and SSIM = 0.99. When the block size changes to 8×8, the results are different. Fig. 7(e) shows PSNR = 23.02 and SSIM = 0.49 on the Baboon image, while Fig. 7(f) shows PSNR = 29.92 and SSIM = 0.90 for the Pepper image. The two images show a difference in performance. However, the quality appearance is not visually noticeable to the observer as shown in Figs. 7(a)-(f). Various block size effects are also shown.

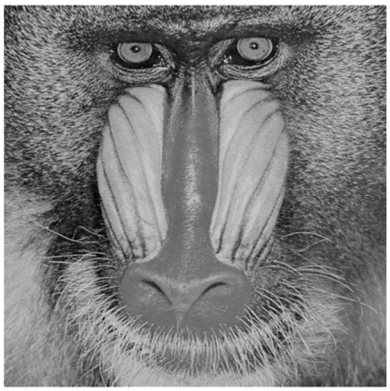


(a) Baboon image (b) Pepper image

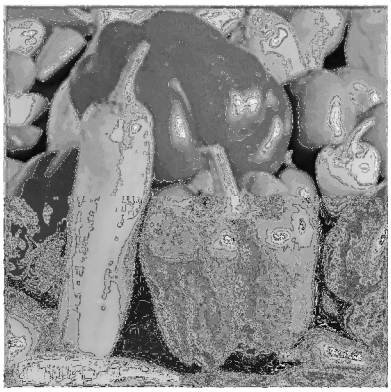
Fig. 6 Two images under test

Universal image quality index (UIQI) is also shown in Tables 2 and 3 for purposes of quality review. The UIQI metric measures the image distortion as a combination of loss of correlation, luminance, and contrast distortion [26-27].

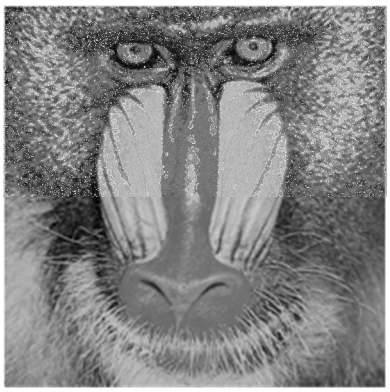
For the purpose of evaluating the proposed method in terms of CR, PSNR, SSIM, UIQI, and execution time, set partitioning in hierarchical trees (SPIHT) and Joint Photographic Group (JPG) algorithms are used. SPIHT is very effective in terms of CR. The level index is used to choose the preferred CR [28-29]. The compression index is calculated as data size (the number of DCT coefficients) divided by block size. Tables 2, 3, and 4 show the proposed method performance compared to the SPIHT and JPG algorithms, respectively, in terms of CR, PSNR, SSIM, UIQI, and execution speed, when applied to the Pepper image. The execution speed depends on the hardware and software being used. The reported results, rather than the absolute values, are shown here for purposes of comparison. Tables 5 and 6 show these metrics when applied to the Baboon image. Fig. 8 compares the image quality after being decompressed and shows the execution time differences between the proposed method and the SPHIT method. It can be seen that the proposed method is at least 10 times faster than the SPIHT algorithm. The SPIHT algorithm outperforms the proposed method in terms of image quality for relatively low CR. When the CR gets higher, the differences between the two methods start to disappear. However, the proposed method is much faster than the SPIHT method.



(a) Baboon uncompressed image

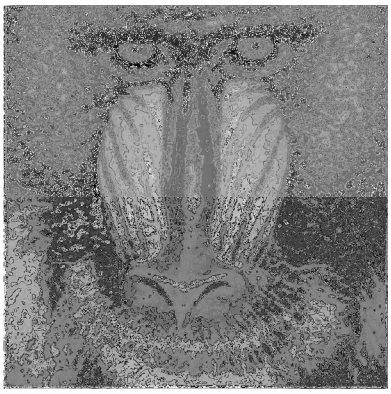


(b) Pepper uncompressed image



(c) Compressed image 3×3 block size

(d) Compressed image 3×3 block size



(e) Compressed image 8×8 block size

(f) Compressed image 8×8 block size

Fig. 7 The effect of the proposed method on image quality for various compression ratio

Table 2 Performance accuracy of the proposed method on Pepper image

Block Size	CR index	CR %	PSNR	SSIM	UIQI	speed
3 × 3	2.25	55	44.25	0.99	0.97	0.0120
4 × 4	4.00	75	39.89	0.98	0.93	0.0113
5 × 5	6.25	84	36.85	0.97	0.88	0.0111
6 × 6	9.00	89	34.18	0.95	0.83	0.0110
7 × 7	12.25	92	32.07	0.93	0.78	0.0117
8 × 8	16.00	94	29.92	0.90	0.73	0.0108

Table 3 Performance accuracy of SPIHT method on Pepper image

SPIHT Index	CR index	CR %	PSNR	SSIM	UIQI	speed
11	4.2	76	44.55	0.99	0.99	0.0942
12	7.5	86	43.55	0.99	0.98	0.1023
13	12.5	92	41.40	0.98	0.95	0.1192
14	19.7	95	38.24	0.97	0.91	0.1401
15	30.0	97	34.57	0.94	0.84	0.1766
16	43.3	98	30.70	0.89	0.75	0.2154

Table 4 Performance accuracy of JPG method on Pepper image

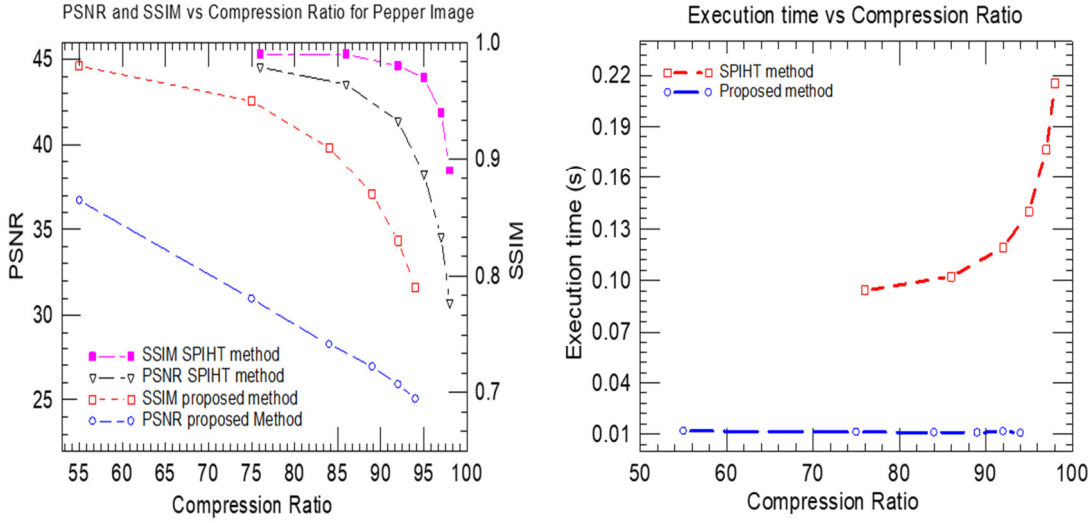
Compression Quality (JPG input)	CR %	PSNR	SSIM	UIQI	speed
100	80	44.84	0.99	0.95	0.0223
90	93	40.48	0.99	0.89	0.0151
80	95	38.29	0.99	0.85	0.0153
70	96	36.99	0.99	0.82	0.0119
60	97	35.84	0.99	0.79	0.0136
30	98	32.66	0.98	0.68	0.0127

Table 5 Performance accuracy of the proposed method on Baboon image

Block Size	CR index	CR %	PSNR	SSIM	UIQI	speed
3 × 3	2.25	55	28.17	0.88	0.88	0.0120
4 × 4	4.00	75	25.80	0.76	0.77	0.0114
5 × 5	6.25	84	24.59	0.67	0.67	0.0113
6 × 6	9.00	89	23.85	0.59	0.59	0.0112
7 × 7	12.25	92	23.32	0.54	0.52	0.0105
8 × 8	16.00	94	23.02	0.49	0.46	0.0155

Table 6 Performance accuracy of SPIHT method on Baboon image

SPIHT Index	CR index	CR %	PSNR	SSIM	UIQI	speed
11	4.2	76	44.46	0.99	0.99	0.1028
12	7.5	86	43.40	0.99	0.99	0.1338
13	12.5	92	40.24	0.98	0.98	0.1750
14	19.7	95	35.52	0.96	0.95	0.2142
15	30.0	97	30.71	0.89	0.86	0.2626
16	43.3	98	26.49	0.74	0.69	0.3105



(a) Quality vs compression ratio (b) Quality vs execution time

Fig. 8 Compression quality and speed of SPIHT vs the proposed method on pepper image

It is important to test the algorithm on many images with various properties, such as texture and entropy. The proposed method shows comparable quality to the JPG and SPIHT methods on some images, but not on others. Also, it shows faster execution time than both methods for some test images. Table 7 and Table 8 show the comparison for two different images, namely, Cameraman and Barbara.

Table 7 Performance accuracy of proposed, JPG and SPIHT methods on Cameraman image

Proposed method			JPG Method			SPIHT method		
CR %	PNSR	Speed	CR %	PNSR	Speed	CR %	PNSR	Speed
84	24.8	0.0113	81	35.8	0.0057	86	40.5	0.159
92	23.0	0.0112	93	29.9	0.0049	92	37.2	0.144
94	22.4	0.0164	95	26.5	0.0048	95	33.1	0.136

Table 8 Performance accuracy of proposed, JPG and SPIHT methods on Barbara images

Proposed method			JPG Method			SPIHT method		
CR %	PNSR	Speed	CR %	PNSR	Speed	CR %	PNSR	Speed
84	22.0	0.0164	81	33.4	0.0110	86	40.8	0.185
92	21.1	0.0176	93	26.8	0.0092	92	36.6	0.152
94	20.9	0.0164	95	24.4	0.0083	95	32.0	0.123

4. Discussion

Almost all compression algorithms introduce some artifacts in the decompressed images. However, it varies from one algorithm to another. Both the JPG and SPIHT algorithms experience artifacts. Since the proposed method does not have many levels of compression, it introduces artifacts in a consistent manner. To compare the proposed method to the SPIHT and JPG algorithms, Fig. 9 shows the comparison in terms of artifact creation at the lowest CR. It is clear that the proposed method does not introduce artifacts that can be visually noticed. The only method that produces visually noticeable artifacts is the SPIHT method. Fig. 10 shows a visual comparison with more sets of test images. Even though the PSNR of the JPG and SPIHT methods are much higher than in the proposed method, the visual appearance of the three different compression methods is almost similar.

(a) Original Image (b) Proposed method image (c) JPG image (d) SPIHT image

Fig. 9 Pepper image with artifacts

(a) Original Images (b) Proposed method (c) JPG method (d) SPIHT method

Fig. 10 Cameraman and Barbara images compared visually

5. Conclusions

The neural network model is built to predict the DCT coefficients of grayscale images. The NN is trained on a vast number of image samples. To compress an image, the image is divided into blocks of size 8×8 or less. The NN is then used to predict the first four DCT coefficients of each block. The proposed NN is of general nature where it is used each time an image

is to be compressed. This makes the computation time of the image compression step is almost negligible since the predefined NN is used to predict the DCT coefficients. The compression process does not need the DCT coefficients computation anymore, as the general NN model is enough. The decompression is done by calculating the inverse of DCT four coefficients, with enough padding with zeros. The resulting quality of the decompressed images is satisfactory. In fact, for low CR, the proposed compression method can reach lossless compression quality. The proposed method outperforms the SPIHT algorithm in terms of execution speed on the compression side. The suitability of using the proposed method in color images is under investigation by the author.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] C. G. Bampis, Z. Li, I. Katsavounidis, T. Y. Huang, C. Ekanadham, and A. C. Bovik, "Towards Perceptually Optimized End-To-End Adaptive Video Streaming," Arxiv E-prints, arXiv:1808.03898v1, 2018.

[2] M. Tawalbeha, A. Eardley, and L. Tawalbeh, "Studying the Energy Consumption in Mobile Devices," Procedia Computer Science, vol. 94, pp. 183-189, 2016.

[3] "Apple X ®," https://www.apple.com/iphone-xr/specs/, April 12, 2020.

[4] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 3rd ed. Upper Saddle River: Pearson Education, Inc., 2008.

[5] K. Pathak, R. V. Arjunan, and V. Acharya, "An Innovative Lossless Image and Video Compression Using Revised S Transformation," Journal of Advanced Research in Dynamical and Control System, vol. 11, no. 4, pp. 14-24, 2019.

[6] S. UmaMaheswari and V. SrinivasaRaghavan, "Lossless Medical Image Compression Algorithm Using Tetrolet Transformation," Journal of Ambient Intelligence and Humanized Computing, in press.

[7] A. M. Raid, W. M. Khedr, M. A. El-Dosuky, and W. Ahmed, "JPEG Image Compression Using Discrete Cosine Transform-A Survey," International Journal of Computer Science & Engineering Survey, vol. 5, no. 2, pp. 39-47, April 2014.

[8] N. Ahmed, "How I Came Up with the Discrete Cosine Transform," Digital Signal Processing, vol. 1, no. 1, pp. 4-5, January 1991.

[9] J. G. Daugman, "Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 36, no. 7, pp. 1169-1179, July 1988.

[10] C. Amerijckx, M. Verleysen, P. Thissen, and J. D. Legat, "Image Compression by Self-Organized Kohonen Maps," IEEE Transactions on Neural Networks, vol. 9, no. 3, pp. 503-507, May 1998.

[11] D. P. Dutta, S. D. Choudhury, M. A. Hussain, and S. Majumder, "Digital Image Compression Using Neural Networks," International Conference on Advances in Computing, Control, and Telecommunication Technologies, December 2009, pp. 116-120.

[12] J. Robinson and V. Kecman, "Combining Support Vector Machine Learning with the Discrete Cosine Transform in Image Compression," IEEE Transactions on Neural Networks, vol. 14, no. 4, pp. 950-958, July 2003.

[13] R. D. Dony and S. Haykin, "Neural Network Approaches to Image Compression," Proceedings of the IEEE, vol. 83, no. 2, pp. 288-303, February 1995.

[14] V. R. P. Vaddella and K. Rama, "Artificial Neural Networks for Compression of Digital Images: A Review," International Journal of Reviews in Computing, vol. 3, pp. 75-82, June 2010.

[15] S. A. Alshehri, "Video Compression Using Frame Redundancy Elimination and Discrete Cosine Transform Coefficient Reduction," Multimedia Tools and Applications, vol. 80, no. 1, pp. 367-381, January 2021.

[16] K. Dimililer, "Backpropagation Neural Network Implementation for Medical Image Compression," Journal of Applied Mathematics, vol. 2013, 453098, December 2013.

[17] B. Patel and S. Agrawal, "Image Compression Techniques Using Artificial Neural Network," International Journal of Advanced Research in Computer Engineering and Technology, vol. 2, no. 10, pp. 2725-2729, October 2013.

[18] V. Mehare and S. Shibu, "A Neural Network Approach to Improve the Lossless Image Compression Ratio," Peoples Journal of Science & Technology, vol. 2, no. 1, pp. 53-58, January-June 2012.

[19] P. Karthikeyan and N. Sreekumar, "A Study on Image Compression with Neural Networks Using Modified Levenberg-Marquardt Method," Global Journal of Computer Science and Technology, vol. 11, no. 3, pp. 1-5, March 2011.
[20] S. Alshehri, "Neural Network Technique for Image Compression," IET Image Processing, vol. 10, no. 3, pp. 222-226, March 2016.
[21] S. Alshehri, "English Characters OCR Pertinent for Mobile Devices," International Journal of Computing and Digital Systems, vol. 10, no. 1, pp. 135-141, 2021.
[22] S. Ma, X. Zhang, C. Jia, Z. Zhao, S. Wang, and S. Wang, "Image and Video Compression with Neural Networks: A Review," IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, no. 6, pp. 1683-1698, June 2020.
[23] G. Schaefer and M. Stich, "UCID: An Uncompressed Color Image Database," Storage and Retrieval Methods and Applications for Multimedia, vol. 5307, pp. 472-480, December 2003.
[24] C. M. Bishop, Neural Networks for Pattern Recognition, New York: Oxford University Press, Inc., 1997.
[25] A. Horé and D. Ziou, "Image Quality Metrics: PSNR vs. SSIM," 20th International Conference on Pattern Recognition, August 2010, pp. 2366-2369.
[26] C. Rawat and S. Meher, "A Hybrid Image Compression Scheme Using DCT and Fractal Image Compression," International Arab Journal of Information Technology, vol. 10, no. 6, pp. 553-562, November 2013.
[27] S. Thayammal and D. Selvathi, "Edge Preserved Image Compression Using Extended Shearlet Transform," Journal of Computer Science, vol. 11, no. 1, pp. 82-88, 2015.
[28] A. Said and W. A. Pearlman, "A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees," IEEE Transactions on Circuits and Systems for Video Technology, vol. 6, no. 3, pp. 243-250, June 1996.
[29] M. Santhi and R. W. Banu, "Enhancing the Color Set Partitioning in Hierarchical Tree (SPIHT) Algorithm Using Correlation Theory," Journal of Computer Science, vol. 7, no. 8, pp. 1204-1211, 2011.