

# QUALITY ANALYSIS OF LOSSY IMAGE COMPRESSION TECHNIQUES

A Thesis

by

BHAVIK PRASHANT BHOIR

Submitted to the College of Graduate Studies  
Texas A&M University-Kingsville  
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2019

Major Subject: Electrical Engineering


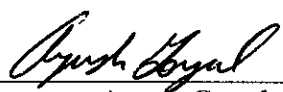
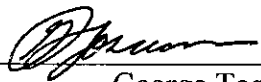

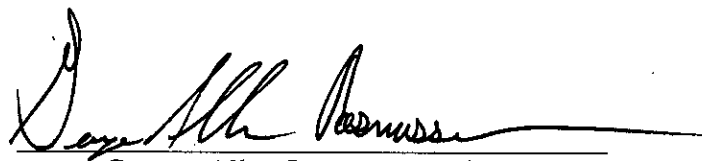
# QUALITY ANALYSIS OF LOSSY IMAGE COMPRESSION TECHNIQUES

A Thesis

by

BHAVIK PRASHANT BHOIR

Approved as to style and content by:

  
Sung-won Park, Ph.D., P.E.  
(Chair of Committee)  
Ayush Goyal, Ph.D.  
(Member of Committee)  
George Toscano, Ph.D.  
(Member of Committee)  
Afzel Noore, Ph.D.  
(Interim Chair of Department)  
George Allen Rasmussen, Ph.D.  
(Vice President for Research and Graduate  
Studies)

May 2019

## ABSTRACT

### Quality Analysis of Lossy Image Compression Techniques

(May 2019)

Bhavik Prashant Bhoir, B.E. University of Mumbai

Chairman of Advisory Committee: Dr. Sung-won Park

In today's digital world, image compression is a crucial aspect of data transmission and storage which has led to an increasing focus on various image compression algorithms. The objective of image compression is the reduction of the number of bits required to store an image and preserve the quality levels of the original image. This research will discuss a few lossy image compression techniques like image compression based on wavelets, image compression based on 00a Feed-Forward Neural Network and two hybrid compressions methods based on the both wavelet functions and Neural Networks. The hybrid technique will compress the input image first by the wavelet-based compression algorithm, and then the output image will be further compressed by a Feed-Forward neural network. Another hybrid technique, in which the image is first compressed using a neural network and then by the wavelet-based algorithm is also analyzed. This research aims at examining how the two proposed hybrid methods perform on various types of images in comparison to the two individual techniques. The compression performance of these techniques in terms of parameters such as compression ratio, peak signal to noise ratio and human visual system based parameters determines the efficiency of these hybrid techniques.

## ACKNOWLEDGEMENTS

The thesis work has been a learning process for me, both on an academic and personal level. And, now that I have reached the finish line, I would like to acknowledge certain people.

Firstly, I wish to express my sincere appreciation and gratitude to my thesis advisor Dr. Sung-Won Park, for his guidance and support throughout my thesis research. His useful insights and motivation helped me get the thesis work done smoothly and within time.

Next, I would like to acknowledge and thank my committee member Dr. Ayush Goyal for his valuable inputs and taking out time for my research work. Furthermore, I would also like to thank Dr. George Toscano for his suggestions and support during my thesis work.

Finally, I would like to convey my thank you to my mom, my aunt, and my friends for their constant support and unconditional love throughout my entire master's journey. This achievement would not have been possible without their support and efforts.

TABLE OF CONTENTS	Page
ABSTRACT .....	iii
ACKNOWLEDGMENTS .....	iv
LIST OF FIGURES .....	vii
LIST OF TABLES .....	viii
CHAPTER I. INTRODUCTION.....	1
1.1 Problem Statement.....	5
1.2 Objective.....	6
CHAPTER II. LITERATURE REVIEW .....	7
CHAPTER III. METHODOLOGY .....	12
3.1 Proposed Work .....	12
3.2 Algorithms .....	12
3.2.1 Wavelet Transform .....	12
3.2.2 Neural Network .....	18
3.2.3 Hybrid Wavelet Transform - Neural Network .....	23
3.2.4 Hybrid Neural Network - Wavelet Transform .....	24
3.3 Performance Parameters .....	25
3.3.1 Compression Ratio .....	26
3.3.2 Mean Square Error.....	26
3.3.3 Peak Signal to Noise ratio .....	26
3.3.4 Peak signal-to-noise ratio based on the human visual system.....	27
3.3.5 Structural Similarity Index .....	31
CHAPTER IV. RESULTS AND DISCUSSIONS .....	34

4.1 Result Analysis A .....	35
4.2 Result Analysis B .....	45
4.3 Result Analysis C .....	55
CHAPTER V. CONCLUSION.....	57
CHAPTER VI. FUTURE SCOPE .....	58
REFERENCES .....	59
APPENDIX.....	62
VITA.....	76

LIST OF FIGURES	Page
Fig. 3.1. One filter stage in 2-D Discrete Wavelet Transform [16, Fig. 3(a)] .....	15
Fig. 3.2. Wavelet decomposition structure [16, Fig. 3(b)].....	15
Fig. 3.3. A general three-layer feed-forward neural network architecture [10, Fig. 4] .....	19
Fig. 3.4. Block Diagram of Hybrid (WT-NN) Image Compression Technique .....	23
Fig. 3.5. Block Diagram of Hybrid (NN-WT) Image Compression Technique .....	24
Fig. 4.1. Image Results of Case 1A [19].....	36
Fig. 4.2. Image Results of Case 2A [20].....	38
Fig. 4.3. Image Results of Case 3A [21].....	40
Fig. 4.4. Image Results of Case 4A [22].....	42
Fig. 4.5. Plot 1 for Comparison of all input images.....	43
Fig. 4.6. Plot 2 for Comparison of all input images.....	44
Fig. 4.7. Image Results of Case 1B [19] .....	46
Fig. 4.8. Image Results of Case 2B [20] .....	48
Fig. 4.9. Image Results of Case 3B [21] .....	50
Fig. 4.10. Image Results of Case 4B [22] .....	52
Fig. 4.11. Plot 3 for Comparison of all input images.....	53
Fig. 4.12. Plot 4 for Comparison of all input images.....	54
Fig. 4.13. Image Results of Case 1C [22].....	56

## LIST OF TABLES

Page

Table 4.1. Parameter Analysis of Case 1A .....	35
Table 4.2. Parameter Analysis of Case 2A .....	37
Table 4.3. Parameter Analysis of Case 3A .....	39
Table 4.4. Parameter Analysis of Case 4A .....	41
Table 4.5. Parameter Analysis of Case 1B .....	45
Table 4.6. Parameter Analysis of Case 2B .....	47
Table 4.7. Parameter Analysis of Case 3B .....	49
Table 4.8. Parameter Analysis of Case 4B .....	51
Table 4.9. Parameter Analysis of Case 1C .....	55



## CHAPTER I

### INTRODUCTION

Advancements in technology have increased the extent of data present in the digital world. Hence, compressing the data has become an essential research area. There are several types of data like images, video, text, etc. Data compression is known as the process that reduces the amount of redundant data while decreasing the transmission costs and the data storage requirements. Compressing the data leads to an increase in the data storage and data transfer capacity of a medium which has increased the focus on the evolution of various compression techniques. Data refers to a representation of the mixture of information and redundancy. Redundancy is something that can be removed or reinserted as per the need. Data compression is a technique used to eliminate redundancy. [1].

Data compression is classified into lossless compression methods and lossy compression methods. As the name suggests, lossless compression methods involve zero information loss. Whereas, in the case of lossless compression, data recovery is possible even after compression. It is also familiar as reversible compression as the original data can be precisely reconstructed even after compression [2]. These methods find use in applications in which even a slight loss of information is intolerable, which suggests that the original and reconstructed data should be similar. There are specific applications, in which we can tolerate some loss of information. In these applications, we turn our focus to lossy compression methods [1]. In lossy compression, usually, data cannot be reconstructed once it is compressed. They are also known as non-reversible compression methods as the reconstructed data and the original information is not the same which gives us greater compression ratios when compared to the lossless compression methods [2]. For certain applications, the insufficiency of reconstruction does not pose an issue. One such use is

storage or transmission of a speech signal where the knowledge of each speech sample's exact value is needless. The tolerance of data depends on the desired reconstructed speech quality. Similarly, in the case of video data, if the reconstructed signal doesn't present any annoying artifacts, the difference between its original and reconstructed form is tolerable. Thus, video compression applications prefer lossy compression methods [1].

The research presented in this work focuses on image compression. An image can be considered as a binary rendition of visual data. There are five main formats for digital images - Tagged image file format (TIFF) which contains extensive image data and follows lossless compression. Joint photographic experts group (JPEG) file format, ideal for detailed images like web, e-mail or multimedia as they have small file sizes and require less loading time. Graphics interchange format (GIF), compressed in a lossless manner, unlike JPEG images find application for images with few colors like animations. Portable network graphics (PNG) format which follows a lossless compression algorithm find a use for web images like screenshots that are a mixture of pictures and text, and raw image files which contain data directly from a digital camera which means they are not processed and non-printable or edited [3]. JPEG has turned out to be an international standard for image compression [4].

Image compression is a classification of data compression on digital images. It aims at reducing the redundancies present in an image and efficiently increase the data storage and transmission ability of a system [5]. The degree of redundancy reduction is analogous to the amount of compression achieved [5]. Like data compression, compression of images can be lossless or lossy. In Lossless image compression, zero information or pixels loss is observed. Hence, the original image can be reconstructed even after compression. This technique is also known as entropy coding because the compressed signal is usually more random as compared to

the original. But, lossless compression techniques fail to offer sufficiently high compression ratios. Applications like medical imaging, legal records, or comics for archival purposes favor lossless compression as they require exact recoveries of original images [2]. There are various types of such techniques which are discussed below.

Run length encoding (RLE): It is a technique that counts the number of values (characters or symbols) to compress string data and then assigns a character in a location along with its count. This technique is ideal in case of grayscale images. For example, a string “aaaabbccdd” will be compressed to get a4b3c3d3 as the output.

RLE is easy to implement and serves as an excellent alternative to the complex compression algorithm because of its high execution speed. However, it fails to achieve a high compression ratio. This technique finds its application for compression of TIFF, PDF and BMP file types [5].

Lempel-Ziv-Welch (LZW): It is a type of lossless image compression technique developed by Terry Welch. It was initially being designed for text compression but has also found application in the field of signal compression [2]. This technique aimed at replacing the data with a look-up table with encoded data during compression and while decoding the data [5]. The primary process in an LZW technique is as follows:

- a. Assign a code to the corresponding characters of an input file.
- b. Assign an existing code if we come across a figure which already exists in the file
- c. Repeat the process until all aspects of the file are null.

LZW find use in compression of TIFF, GIF, and PDF file types. In this technique, the management of the string table becomes an issue as it only works on English text and needs a dictionary.

Entropy encoding: It is a type of lossless image compression technique that operates solitarily irrespective of the image attributes. This technique is useful to achieve more compression by using the probability mass function (PMF) once the data is decorrelated [2]. It allocates a unique prefix code to every single input symbol to measure the similarity in the data. This technique compresses data by substituting a prefix for the fixed length output, unlike RLE. Huffman coding and Arithmetic coding are the two widely used entropy coding techniques. Arithmetic coding is useful in cases of a symbol by symbol encoding. In this type of coding, codewords get generated by partitioning the range of numbers between zero and one [2]. Huffman coding encodes a symbol or character in a file based on a code table with variable length. This technique can be implemented easily and has a high execution time. It finds its application in case of compression of ZIP, MPEG, JPEG, ARJ file types [5].

Lossy image compression techniques involve some loss of information which means exact reconstruction is not possible. When compression is performed at small bit rates, this technique tends to introduce some distortions. They find a preference in applications which involve images in which little loss of accuracy is tolerable to obtain a low bit rate. One such example is an image of a tree, which occupies a significantly high amount of memory space. Lossy compression will drastically reduce the file size while getting rid of some minute details which are usually acceptable.

In some cases, lossy compression produces negligible differences and hence can tend to be visually lossless. These techniques are favored in multimedia applications like video conferencing

to achieve increased compression performance with an acceptable level of error. There are various types of lossy image compression techniques that are discussed below.

**Transform coding:** Transform coding techniques are usually applied in lossy compression systems as they possess excellent decorrelation abilities [2]. This technique is known for its ability to produce higher compression based on significantly large computations. This technique transforms the spatial domain depiction of an image to a new, distinct depiction by applying various transforms and then performs coding on the transformed coefficients [5]. One most common example of this type is the Discrete Cosine Transform (DCT). It was developed to produce a finer approximation of the signals with minimal coefficients and is compatible with computation based on Fast Fourier Transform (FFT) [6]. DCT for lossless compression systems is not acceptable as the transform coefficients of DCT are either real-valued or involved valued and require quantization for coding [2].

**Vector coding:** The vector quantization has two sections – encoder and decoder. The encoder produces an output index of the codeword with the least error based the applied input vector. The error is measured as the Euclidean distance between the applied input vector and each codeword present in the designed codebook. The nearest interpretation of the codeword is determined and sent through a transmission channel. On the receiver side, this index gets replaced with an associated codeword by the encoder [5].

### 1.1. Problem Statement

In this digital world, there are millions of images floating around various digital platforms. Thus, there needs to be a way to efficiently compress these images while maintaining their quality to increase the storage capacity of devices and the transmission bandwidth. Many algorithms

developed for this purpose, possess their limitations as they do not perform well when analyzed based on human perception. This research attempts to create an algorithm that overcomes these problems.

## 1.2. Objective

The study aims at developing an efficient hybrid technique which combines the advantages of two conventional lossy image compression techniques to obtain better image compression. The results will be evaluated based on specific performance parameters in comparison to traditional methods.

## CHAPTER II

### LITERATURE REVIEW

Image compression finds a vital part in different image processing applications which has shifted the focus of various researchers on the development of various high-efficiency image compression techniques. A technique is efficient if it satisfies the various performance parameters. Many authors have researched in this field. Gaurav Kumar and Pradeep Kumar Bhatia analyzed image compression using wavelets, Discrete Cosine Transform (DCT) and Neural Networks [6]. They compared these techniques in terms of specific performance parameters like peak signal to noise ratio, output image size, and retained energy. They analyzed that wavelet compression provides optimum compression ratio, MSE, PSNR, and retained energy. It was implemented using wavelets with different decomposition levels. DCT implementation divides the input image into blocks of half, quarter or eight parts. An artificial neural network carries out the compression and decompression of the input image. A cascade-forward backpropagation network was made ready for training. The image is converted into blocks, and the complexity of each obtained block was computed. In conclusion, on paper, the image compression based on wavelets produced better results, but from the empirical results, it was dependent on various filters and levels of decomposition. DCT is much based on the block size while the neural network was found to be best suited for compression of grayscale images [6].

A.S Lewis and G. Knowles discussed image compression based on 2-D wavelet transform. They discussed how images are decomposed into both spatial and spectral coefficients using the 2-D orthogonal wavelet transform [7]. They presented that the transform coefficients can be coded and separately quantized under the human visual system. They proposed that this technique is mappable onto a VLSI architecture.

As per [8], Alok Kumar Singh and G.S. Tripathi conducted an extensive analysis of image compression based on DCT, DWT, and a proposed hybrid DCT-DWT technique. They performed a comparative analysis of the three techniques and found that the DCT-DWT method produces better results than the individual JPEG-based DCT and DWT techniques in terms of parameters like PSNR and visual perception at greater compression rates that determine the efficiency of compression techniques. The results showed that DWT produced higher compression ratio than DCT. However, DWT tends to take up more processing power. DCT produces blocking artifacts at smaller bit rates. On the other hand, the hybrid technique produced significantly higher compression rates than the other techniques while preserving the image information and produce better reconstruction while image clarity being a trade-off. The hybrid technique reduced general issues such as blocking artifacts, false contouring and ringing effect [8].

In recent years, with so much development in the digital world, neural networks have come under significant focus. They find their applications in cognitive tasks like image recognition, image compression, natural language understanding, and many more.

Several algorithms designed for image compression produce nicely compressed images while maintaining their quality. However, these tend to be quite slow as they fail to perform parallel execution over multiple CPU or GPU cores. Image compression using neural networks perform better in comparison to other algorithms and even eliminate the common issues of parallel computing and processing time. Hence, many image compression techniques using neural networks came into consideration. Various authors researched this topic using different neural network architectures.

Authors Rudy Setiono, Guojun Lu presented a feed-forward neural network image compression algorithm with a single hidden layer. The algorithm was designed in such a manner



that any addition to the hidden layer of the network, improves the image quality making the algorithm flexible and presented the user with an option to choose between compression ratio and visual image quality as the application parameters [9].

An image compression method based on neural networks and wavelet functions combined the advantages of the two conventional techniques: wavelet transform and neural network [10]. Wavelet filters were used to decompose the input images into sub-images of varying resolutions corresponding to the several bands. Various quantization and coding approaches were adopted for different sub-bands. The low-frequency band coefficients were compressed using Differential pulse code modulation (DPCM) and a neural network was used to compress the higher band coefficients. One level of image decomposition produced seven bands. Band one was encoded using DPCM, and then the coefficients were scalar quantized. The same neural network compressed the data in bands two and three as they consisted of same frequency data while a different neural network was used to compress data in bands five and. Band four components were compressed using a completely different neural network as they have different frequency components than the other bands. Band seven contains very little information to affect the reconstruction quality, and hence this band is discarded. The hidden layer outputs are scalar quantized and then entropy encoded by Huffman encoding. This technique produced a satisfactory reconstructed image with significantly higher compression ratios. DAUB 18 produced better reconstruction filters in comparison to other wavelet filters used for experimentation [10].

A digital image compression technique using deep neural networks was implemented [11]. They used two different DNN architectures: logistic sigmoid and hyperbolic tangent which differ in terms of the activation function (neurons). DNNs are artificial neural networks (ANN) consisting of several hidden layers between the input and output layers which depict a broader set

of functions. The two schemes were compared based on peak signal-to-noise of the images by adjusting the input, output, and hidden layer neurons to obtain several compression rates. Comparison of the epoch vs. MSE curves of the employed two schemes showed that the hyperbolic tangent network produces a higher convergence rate. Results indicated that hyperbolic tangent neural network performs better than the sigmoid neurons as they produce improved PSNR for reconstructed images and a faster magnitude than the logistic sigmoid neurons which means that the neurons train faster than the ones in the sigmoid scheme. This algorithm tends to provide a compromise between the quantity of compression and image quality. It was also proposed that there may be a future scope that focuses on GPU implementation of the discussed networks [11].

A new approach to the image compression problem was later proposed which focused on a hybrid image compression method using Neural Networks and Wavelets [12]. This method aimed at combining the advantages of the two standalone techniques. This scheme produced high compression rates and possessed the recreation properties of wavelets. SOFM algorithm produced better compression ratios amongst the various neural networks. This method focused on compressing grayscale images. The hybrid method used SOFM neural network to obtain code vectors and then compressed them again by DWT algorithm. The research compared this hybrid method to conventional SOFM neural network algorithm in terms of statistical parameters like compression ratio, PSNR and some co-relational metrics like average difference and image fidelity. Implementation was carried out using various clusters such as 64,128 and 256. The outcome indicated that the hybrid method produced better image compression results when compared to the conventional method [12].

Many pieces of research have thus been carried out focusing on the development of various hybrid image compression methods. Google has always made progress in computer vision filed

and now focused on various techniques involving machine learning for image compression. One such is presented in [13] by authors George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor and Michele. The objective was to create a neural network which is productive for variable compression ratios on arbitrary size images. The architecture consists of an encoder and decoder based on recurrent neural network (RNN). They pointed out for evaluation of this architecture, reliance on normal performance parameters like PSNR is not acceptable as the human visual system tends to possess higher sensitivity to certain types of errors [14] [15]. They selected PSNR based on the Human Visual System discussed in [14] and multiscale structural similarity index discussed in [15] for evaluating the performance. The architecture performed better than the conventional JPEG system by providing considerably higher PSNR-HVS and MS-SSIM both with and without entropy coding [13]. They also discussed a future challenge relating to this research which focused on certain video compression formats like WebP to be used for compression significantly large images. They also discussed a future scope for a single perceptual image quality evaluation metric which eliminates the limitations of the current metrics.

## CHAPTER III

### METHODOLOGY

#### 3.1. Proposed work

Different lossy image compression methods such as image compression using discrete wavelet transform, image compression using a feed-forward neural network and two hybrid techniques using both discrete wavelet transform and a feed-forward neural network are analyzed. The compression performance of these methods in terms of parameters such as Compression ratio, Peak signal to noise ratio, and Human Visual System based parameters determines the efficiency of these hybrid techniques.

#### 3.2. Algorithms

##### 3.2.1. Wavelet Transform

In recent years, the discrete wavelet transform for compressing images has gained much interest in the research community. Wavelets are known as signals with irregular shape and zero average value [5]. In Fourier series representation, the time resolution is generally not good. Whereas, in the case of wavelet representation, signals represent functions which are both frequency and time-localized. Wavelets tend to have an adaptive time-frequency window and hence, are efficient in representing non-stationary signals. They seem to possess high decorrelation efficiency along with good energy compaction productivity. Wavelets tend to automatically adapt to both frequency components present in high and low regions of an input signal by varying sized windows [6]. A mother wavelet can result in the construction of a family of wavelets. In, Windowed Fourier analysis, the window size can be altered by stretching or compressing the mother wavelet. Hence, significant wavelets tend to give an approximation of the input, whereas,

the details are determined by the smaller wavelets. Local errors don't impact the complete transform as any minor variation in the wavelet representation will produce a minor variation in the corresponding input. Hence, non-stationary signals prefer wavelet transform [8].

Wavelet transformation calculates the discrete data [5]. It is considered as a possible representation of the image data while not maintaining the storage requirements. For image compression, we need to decide on the coefficients that need to send, and the quantity of bits required to code them [7]. Wavelets aim at the analysis of the input signal at various resolutions, referred to as multi-resolution [8]. In wavelet analysis, the image information is splitted into approximation and detail images [5]. An image coder based on wavelets eliminates the blocking artifacts and mosquito noise. DWT uses a more appropriate set of functions for illustrating the sharp edges in comparison to the cosines in DCT [1]. 2-D Wavelet analysis represents an image as a matrix of M rows and N columns. A waveform pair is used for any level wavelet decomposition of the input image. One represents the high-frequency associated with the detailed images known as wavelet function, while, the other one is for low frequency associated with the approximation of the image that is the scaling function. The wavelet function can be computed as a product of wavelet function  $\Psi(i,c)$  and scaling function  $\Phi(i,c)$  [8]. The DWT of an input signal  $i$  is calculated by passing it through a series of filters (low-pass and high-pass). The samples are initially passed through a low pass filter having an impulse response denoted by  $r$  which results in a convolution of them considering a shift parameter  $k$ .

$$c[n] = (i * r)[n] = \sum_{k=-\infty}^{\infty} i[k]r[n-k] \quad (3.1)$$

A high-pass filter  $h$  is used to decompose the signal simultaneously. The outputs are the detail coefficients (obtained from the high-pass filter) and approximation coefficients (obtained

from the low-pass). The low-pass filter output  $r$  is then subsampled by 2 and then passed yet again through a new low-pass filter  $r$  and a high-pass filter  $h$  with a cut-off frequency which is half of the preceding one.

$$c_{low}[n] = \sum_{k=-\infty}^{\infty} i[k]r[2n-k] \quad (3.2)$$

$$c_{high}[n] = \sum_{k=-\infty}^{\infty} i[k]h[2n-k] \quad (3.3)$$

This decomposition halves the time resolution as the signal is denoted by only half of the output of each filter. However, this doubles the frequency resolution as each output has half the input frequency band. So, the above equations can be expressed concisely by considering an operator for subsampling  $(c \downarrow k)[n] = c[kn]$  as,

$$c_{low} = (i * g) \downarrow 2 \quad (3.4)$$

$$c_{high} = (i * h) \downarrow 2 \quad (3.5)$$

The frequency resolution can be further improved by repeating the decomposition process, and the approximation coefficients are obtained on decomposition using the corresponding filters and then down-sampled. Higher decomposition levels better resolve the significant DWT coefficients from lesser essential coefficients [16].

Fig. 3.1. Shows a two-dimensional discrete wavelet transform for image compression using one filter stage. The subscript indicates the type of filter used to obtain both the horizontal and vertical components [16]. At the output, we have four different components with all achievable combinations of the filters in two directions [16]. An image decomposes into four bands to obtain

an approximation image (LL) and three different detail images for each resolution: diagonal (HH), horizontal (HL), and vertical (LH).

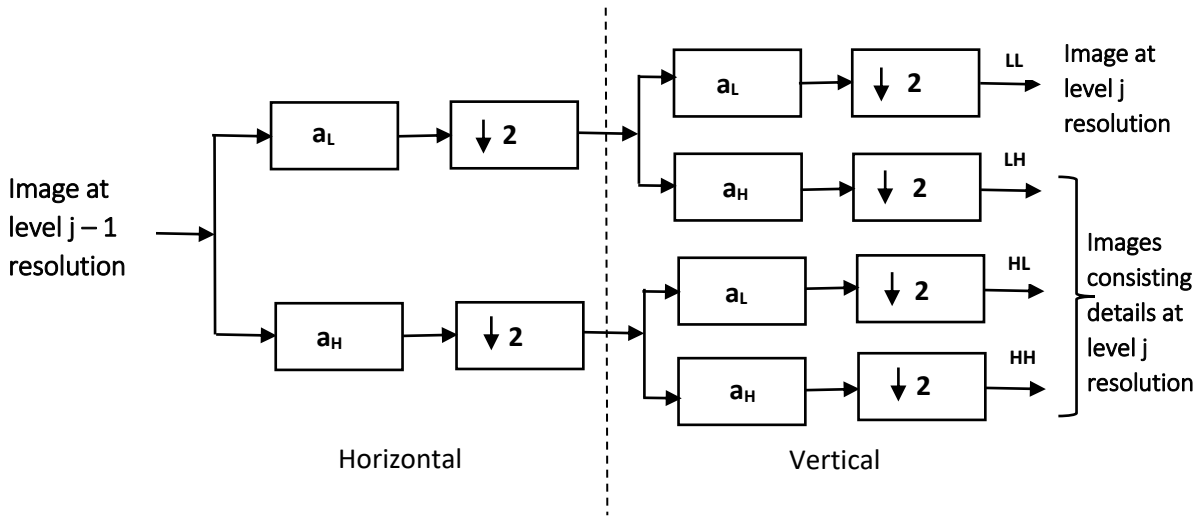


Fig. 3.1. Two-dimensional Discrete Wavelet Transform using one filter stage [16, Fig. 3 (a)]

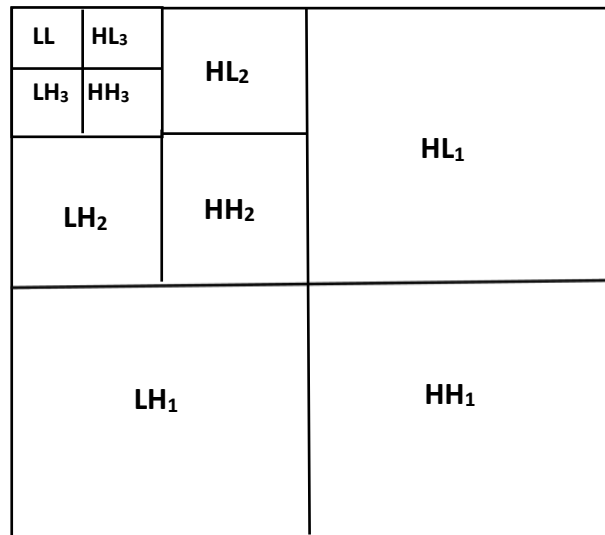


Fig. 3.2. Level 3 Wavelet decomposition structure [16, Fig. 3 (b)]

The approximation image is again applied as an input to the filters to obtain level 2 approximation and detail images. Level 2 approximation image is then again applied to the filters to get level three approximation and detail images. This 2-D discrete wavelet transform generates

a structure which can be seen in Fig. 3.2. In Fig. 3.1,  $j$  represents the number of decompositions which denote the amount of the 2-dimensional filter stages applied for the image decomposition process [16]. At each level, four different sub-images as displayed in Fig. 3.1 get formed. Here, LL denotes the approximation image of decomposition at level  $j$  which is a result of low-pass filtering in both vertical and horizontal directions. HL denotes the horizontal details of  $j$ th level decomposition resulting from horizontal low-pass filtering and vertical high-pass filtering. LH denotes the vertical details of  $j$ th level decomposition resulting from vertical low-pass filtering and horizontal high-pass filtering. HH denotes the diagonal details of  $j$ th level decomposition resulting from high-pass filtering in both vertical and horizontal directions [16]. Fig 3.2 displays a level 3 wavelet decomposition structure.

Selecting a proper wavelet function is essential to achieve optimal image compression [16]. The input image's spectral activity determines how a wavelet will affect the performance of the compression. For instance, an image with greater spectral activity will not be that changed by the selection wavelet when compared to an image with low spectral activity. This research will focus on two wavelets from various wavelet families. Daubechies and Coiflet wavelets have compact support as they are a part of orthogonal wavelet families. However, these two wavelets possess a significant disadvantage which is asymmetry which results in artifacts near the borders of wavelet subbands. In the case of wavelets, the user must choose between having symmetry or having either small support or orthogonality. Haar wavelet, on the other hand, is orthogonal, symmetric and compactly supported. Biorthogonal wavelets have both symmetry and compact support while sacrificing orthogonality but display linear phase attribute. They use two wavelets, one of which is used for decomposition and the other one is used for synthesis instead of the only one wavelet.



For research, a bior4.4 wavelet is used in which the 4 on the left side indicates wavelet for decomposition and 4 on the right side demonstrates reconstruction.

The overall process can be summarized as,

1. Read Input Image O.
2. Perform 2-D wavelet decomposition at level L using the Matlab function wavedec2 as,

$$[c,l] = \text{wavedec2}(O,L,\text{wavelet})$$

This returns a wavelet decomposition of matrix O at level L using wavelet selected by the user. The wavelet decides the decomposition low pass filter and high pass filter. The Outputs obtained are a decomposition vector c and an associating bookkeeping matrix l. The vector c is arranged as a vector with the approximation and detail coefficients at every level, each of which is a row vector. Matrix l gives us the size of all coefficients. L is the number of decompositions. Increasing the decomposition levels results in greater compression rates. The level of decomposition determines the lowest level resolution in the wavelet domain.

3. Compute the matrices of the coefficients of level L based on [c,l] by using the Matlab command wrcoef2 as,

$$\text{wrcoef2}(\text{'type'}, c, l, \text{wavelet}, L)$$

Where, type = a, h, v, d denotes the type of coefficients.

4. The input image O is then rescaled as integers by using Matlab function wcodemat in terms of integers.
5. The default values for image compression are then determined for discrete wavelet to transform by using the Matlab function ddencmp as,

$$[\text{thres}, \text{sorh}, \text{nkeep}] = \text{ddencmp}(\text{'cmp'}, \text{'wv'}, \text{O})$$

Where 'cmp' denotes compression of image O, and 'wv' denotes the wavelet to be used. The 2D compression based on wavelets is then performed by using the Matlab command `wdencomp` as,

$$[\text{Oc}, \text{COc}, \text{LOc}, \text{Perf0}, \text{Perf12}] = \text{wdencomp}(\text{'gbl'}, \text{c}, \text{l}, \text{wavelet}, \text{L}, \text{thres}, \text{sorh}, \text{nkeep})$$

Where Oc represents the compressed version of Input Image O obtained by global thresholding. The output arguments [COc, LOc] denotes a wavelet decomposition structure of the compressed image Oc. sorh indicates soft ('s') or hard ('h') thresholding. If nkeep = 1, it denotes that the approximation coefficients are kept. Thresholding modifies the coefficients to generate more zeros. In hard thresholding, any coefficient below the selected threshold thres is set to the value zero. This results in the generation of numerous consecutive zeros which take up much less space and can be transmitted more quickly. The perf0 indicates the number of zeros, while, perf12 denotes L2 -norm recovery. Global thresholding indicates thresholding of each sub-image with the exactly similar threshold value.

6. The compressed image Oc is then displayed with the associated input image O.

### 3.2.2. Neural Networks

Neural networks are known for providing higher performance in the image processing domain [6]. Neural networks have come under heavy consideration for image compression because of their ability to offer parallel processing of image data and automatic network training for varying data [10]. Neural Networks are known as integrated adaptive systems as they possess the ability to handle nonstationary details in image data. A neural network is known to be a nonlinearity complex network system [17]. Various neural network architectures are based on mathematical computations and a set of parameters to determine the output. This research focuses on image

compression using a feed-forward neural network with an input layer, an output layer and a hidden layer connecting the two layers. Every layer accepts neurons only from the preceding layer output while adjusting the weight value and network structure to perform nonlinear mapping of the input and output layer [17]. The input to this network is the original uncompressed image data while the output side displays the distorted image data which approximates the original data [9].

A general three-layer feed-forward neural network structure is shown in Fig. 3.4. This network is created using three distinct layers, one input layer, one output layer, and a hidden layer. The hidden layer is present between both the input layer and the output layer and connects the input layer to the output layer. Appropriate selection of the number of neurons  $K$  at the hidden layer is required to achieve compression. These must be selected in such a way that they are less than both input and output layer neurons.

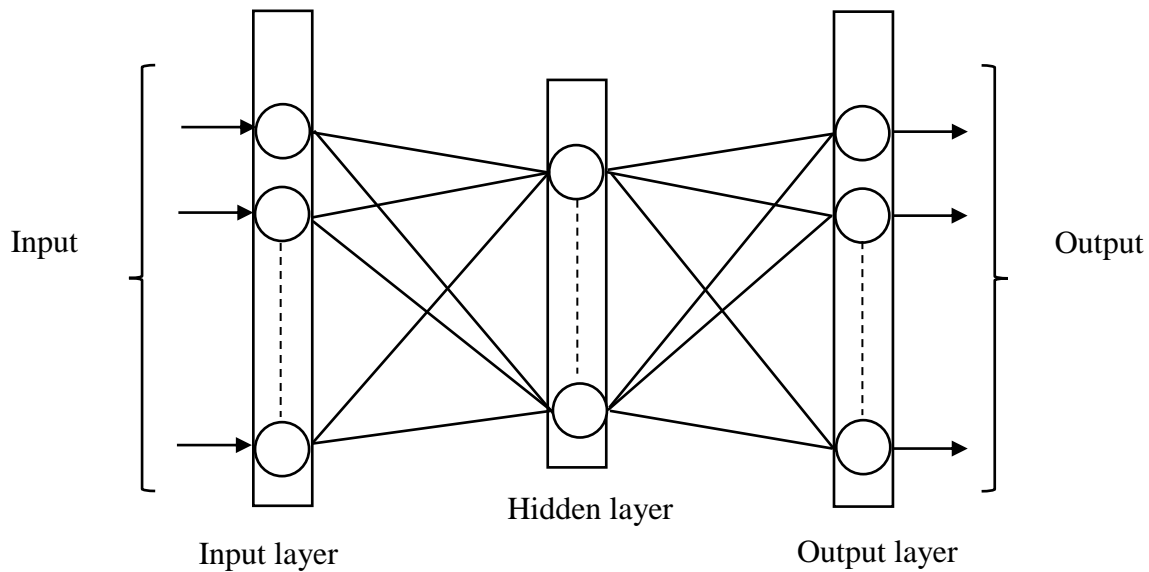


Fig. 3.3. A general three-layer feed-forward neural network architecture [10, Fig. 4]

The input image is divided into blocks or vectors of  $r \times c$  pixel size. The input vector referred to as  $G$ , is attached to each hidden layer neuron and can be indicated by,  $\{w_{ji}; j=1,2,\dots K \text{ and}$

$i=1,2,.. G\}$ . It can be expressed as a matrix of  $K \times G$  dimension. The hidden layer to the output layer connections can be indicated by,  $\{w_{ij}; i=1,2,...,G, j=1,2,...,K\}$ , which denotes a weight matrix of  $G \times K$  dimension. Image compression can be carried by training the created network in a fashion that the associated weights  $\{w_{ji}\}$ , scale the corresponding input vector of  $G$ -dimension into a  $K$ -dimension ( $K < G$ ) channel at the hidden layer and produce minimal error between the input and output.

Image compression based on a feed-forward neural network first divides an image,  $O$ , into blocks of pixels of size  $r \times c$ . These blocks are then scanned to obtain an input vector  $o(n)$  of size  $a=r \times c$ . A weight matrix  $W_h = w_{ji}$  denotes the hidden layer. All  $G$  blocks of the input image are then given to the hidden layer to produce the hidden signals,  $h(n)$ , which indicates the encoded input image blocks. At the output layer, we have  $m=a=r \times c$  neurons with a  $W_c=w_{ij}$ , output weight matrix. All  $G$  hidden vector  $h(n)$ , representing an image  $H$ , is then applied to the output layer to produce the output signal,  $c(n)$ . Reassembly of the output signals into  $a=r \times c$  blocks of the image is then performed. The feed-forward network operates based on the Levenberg – Marquardt backpropagation algorithm. A back-propagation algorithm filters the error information back across the system to adjust the connections between the layers and obtain improved performance. It is also known as a form of supervised learning [10].

The Levenberg Marquardt algorithm is based on Newton's method and aims at minimizing the error functions that are expressed as the sums of squares [17]. In MATLAB, `trainlm` function updates the weight and bias values according to the Levenberg Marquardt algorithm. This algorithm uses Hyperbolic tangent sigmoid transfer function (`tansig`) denoted by  $v$  ( $f(v) = 2/(1+\exp(-2*r))-1$ ) to calculate the output of a node, or neuron, applied as an input. This output is

then applied as input for the next node and so on until an optimal solution to the original issue is obtained.

The performance index  $P(o)$  can be optimized by using Newton's method as,

$$O_{k+1} = O_k - F_k^{-1} g_k \quad (3.6)$$

Where  $F_k = \nabla^2 P(o)$  and  $g_k = \nabla P(o)$ ;

$P(o)$  is assumed as a function of the sum of squares:

$$P(o) = \sum_{r=1}^n v_r^2(o) = v^T(o) v(o) \quad (3.7)$$

The  $j$ th element gradient is expressed as,

$$[\nabla P(o)]_j = 2 \sum_{i=1}^n v_i(o) \delta v_i(o) / \delta o_j \quad (3.8)$$

In matrix form as,

$$\nabla P(o) = 2 J^T(o) v(o) \quad (3.9)$$

Where  $J(o)$  denotes the Jacobian matrix.

Now consider the Hessian matrix, the  $k,j$  element can be expressed as,

$$[\nabla^2 P(o)]_{kj} = \delta^2 P(o) / \delta o_k \delta o_j \quad (3.10)$$

In matrix form,

$$\nabla^2 P(o) = 2 J^T(o) J(o) + 2 S(o) \quad (3.11)$$

Where,

$$S(o) = \sum_{i=1}^n v_i(o) \cdot \nabla^2 v_i(o) \quad (3.12)$$

Considering  $S(o)$  to be small, the approximation of the above matrix can be given as,

$$\nabla^2 P(o) \equiv 2 J^T(o) J(o) \quad (3.13)$$

The Gauss-Newton method is obtained by substituting the values of  $\nabla^2 P(o)$  &  $\nabla P(o)$  in (3.6),

$$O_{k+1} = O_k - [J^T(O_k) J(O_k)]^{-1} J^T(O_k) v(O_k) \quad (3.14)$$

The matrix  $H=J^T J$  may not be invertible. The solution to this issue is expressing the matrix as,  $G = H + \mu A$ . This presents us the Levenberg –Marquardt algorithm which can be expressed as,

$$O_{k+1} = O_k - [J^T(O_k) J(O_k) + \mu_k A]^{-1} J^T(O_k) v(O_k) \quad (3.15)$$

The iterations of the Levenberg- Marquardt backpropagation algorithm (LMBP) are used to minimize the error [17]. For this, all inputs are first applied to the created network and, the associated network outputs and errors over  $P(o)$  are computed based on the transfer function and the parameters of the backpropagation algorithm. Next, the Jacobian matrix is calculated. Then  $\Delta O_k$  is obtained to again calculate the errors using  $O_k + \Delta O_k$ , where,  $\Delta O_k = O_{k+1} - O_k$ . Now, if this newly calculated error is smaller than the earlier calculated error then  $\mu$  is divided by  $v$ , while, considering  $O_{k+1} = O_k + \Delta O_k$ , return to computing the squared errors. If the error is not minimized, then  $\mu$  is multiplied by  $v$  to obtain  $\Delta O_k$  and the process is repeated.

The process of image compression based on a feed-forward neural network of an input image  $I$  of  $R_w \times C_l$  size used as an input can be summarized as,

1. The block matrix  $I$  is first converted into a vector matrix  $V$  of size  $a \times b$  consisting of all the training vectors,  $v(n)$ , obtained from the input image blocks where  $a = r \times c$  and  $b = (Rw \times Cl)/a$ .
2. These vectors are then applied to their respective neurons at the input layer.
3. The training is performed using the Levenberg Marquardt Algorithm which is characterized by the individual weight matrices  $w_{ji}$  and  $w_{ij}$ .
4. The vectors are converted back into blocks of size  $r \times c$  to assemble the output image of size  $Rw \times Cl$ .
5. The designed network is simulated until the calculated error, is considerably minimal or the performance goal is reached.

### 3.2.3. Hybrid Wavelet Transform - Neural Network Technique

This technique aims at combining the two conventional image compression techniques to gain better compression which will be measured based on the performance parameters discussed in section 3.3. The process followed to achieve compression by this method is as shown in Fig. 3.4.

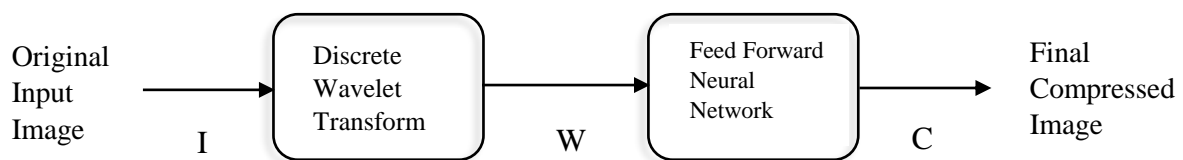


Fig. 3.4. Block Diagram of Hybrid (WT-NN) Image Compression Technique

Here,  $I$  indicates the original input image,  $W$  indicates image compressed using DWT and  $C$  indicates the final compressed image.

The Process presented in the above figure can be summarized as,

1. The input image (I) is compressed using a discrete wavelet transform algorithm as previously explained in section 3.2.1.
2. The compressed image (W) from step 1 is then applied to the designed feed-forward neural network.
3. The image is then compressed by using the neural network algorithm as previously explained in section 3.2.3.
4. The image at the output of the neural network is the final compressed image (C).

The images compressed by discrete wavelet transform and neural network individually will be compared to the final compressed image achieved by this hybrid method based on performance parameters discussed in section 3.3.

#### 3.2.4. Hybrid Neural Network - Wavelet Transform Technique

This technique like discussed in the previous section aims at combining the two conventional image compression techniques to gain better compression which will be measured based on the performance parameters.

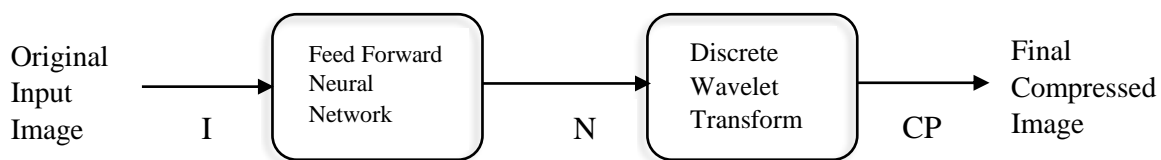


Fig. 3.5. Block Diagram of Hybrid (NN-WT) Image Compression Technique

Here, I indicates the original input image, N indicates the image compressed using Neural Network and CP indicates the final compressed image.

The Process presented in the above figure can be summarized as,



1. The input image (I) is applied to the designed feed-forward neural network and the network is trained as previously explained in section 3.2.3.
2. Image N from step 1 is then further compressed by the discrete wavelet transform algorithm as discussed in section 3.2.1.
3. The image (CP) at the output after step 2 is the final compressed image using the hybrid technique.

The images compressed by the neural network and discrete wavelet transform algorithms individually will be compared to the final compressed image achieved by the hybrid method based on performance parameters discussed in section 3.3.

### 3.3. Performance Parameters

In image processing applications, there are several processes which introduce errors in an image and hence assessment of the quality of an image is a crucial task [14]. Once the image data is compressed, the quality of compression can be measured by using certain quality parameters. They determine how well an image is compressed concerning the original image.

Two quality metrics are generally used in image compression to evaluate compression quality – objective and subjective evaluations. Subjective metrics evaluate an image quality based on the human visual ability and observation while objective metrics evaluate based on mathematical algorithms. Subjective metrics give better results, but they are influenced by personal outlook and physical conditions while analyzing the data [14]. Whereas, objective metrics are widely used because of its simplicity. These metrics are independent of human perspective and physical parameters while being inconsistent with the HVS model [14]. Objective metrics include Compression Ratio (CR), Mean square error (MSE), and Peak signal to noise ratio (PSNR).

### 3.3.1. Compression Ratio (CR)

The ratio between the uncompressed image file size and compressed image file size determines the Compression Ratio. It is used to quantify the achieved compression. Higher compression ratio indicates that more compression is performed.

Compression Ratio can be calculated as,

$$CR = \frac{\text{uncompressed image file size}}{\text{compressed image file size}} \quad (3.16)$$

### 3.3.2. Mean Square Error (MSE)

MSE is a measure of image quality index. Large MSE values imply that the image quality is poor and vice versa [1]. MSE is calculated as,

$$MSE = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [I(x, y) - C(x, y)]^2 \quad (3.17)$$

Here,  $M$  and  $N$  denote the height and width of the image respectively,  $I(x,y)$  denotes the original input image,  $C(x,y)$  denotes the compressed image with  $x$  and  $y$  indicating the pixel positions of the image of size  $M \times N$  [11].

### 3.3.3. Peak Signal to noise ratio (PSNR)

PSNR characterizes the amount of distortion presented by the compression process. The formula used to calculate PSNR is,

$$PSNR = 10 \log_{10} \left( \frac{MAX^2}{MSE} \right) dB \quad (3.18)$$

Here, MAX denotes the maximum possible pixel value (255) of the original input image, while, MSE is the mean square error as discussed in section 3.3.2.

PSNR, as the name suggests, is a relation between the power of the noise that affects the depiction and power of the corresponding signal. The PSNR is presented in decibel as several signals tend to have an extended dynamic range.

Apart from these conventional quality parameters, we will also be considering two other parameters: Peak signal-to-noise ratio based on the human visual system ( $\text{PSNR}_q$ ) and Multi-scale structural similarity index (MS-SSIM). In both metrics, a higher value indicates a higher similarity between the two images under consideration.

The performance of the algorithms needs assessment, and for that, we use two perceptual, full-reference image metrics which compare the original images to the compressed images. It is noteworthy that there is no accord in this domain about which metric is the best representation of human perception, so, it is better to utilize all the available quality metrics while keeping in mind that all metrics possess their strengths and weaknesses [14] [15].

#### 3.3.4. Peak signal-to-noise ratio based on the human visual system ( $\text{PSNR}_q$ )

PSNR fails to produce results that can be similarly identified by the human visual system and hence a modified PSNR metric based on human visual system characteristics is proposed. It considers various parameters for assessing the image [14]. The proposed full-reference image quality metric  $\text{PSNR}_q$  evaluates the weighted sum of error-sensitivity, structural and edge distortions. These distortions are computed for each of the Red (R), Green (G), and Blue (B) color components independently. The process is explained in detail below as presented in [14].

##### a. Error sensitivity measurement

In a color image, the quality depends on the pixel values of the color components of the image. So, any distortion present in the image changes the value of the pixels which are observed by the human visual system. Thus, for quality evaluation, error for each color component is calculated individually.

The error in the red component can be computed as,

$$E_r = \frac{1}{MN} \sum_{x=1}^M \sum_{y=1}^N [I(x, y) - C(x, y)]^2 \quad (3.19)$$

Here,  $I(x, y)$  is the original image, and  $C(x, y)$  is the compressed image,  $x$  and  $y$  are the pixel positions of  $M$  multiplied by  $N$  image.

Similarly,  $E_g$  and  $E_b$  can be calculated for the green and blue components.

The comprehensive error present in the compressed image is calculated as an estimation of the average of the error present in each of the three color components.

The overall error sensitivity can be calculated as,

$$PSNR_{es} = 10 \log_{10} \left[ \frac{3}{E_g + E_r + E_b} \right] \quad (3.20)$$

#### b. Structural distortion measurement

Structural distortion is an essential aspect while assessing the image quality as human eyes usually look at the structural variations in an image.

This type of distortion is calculated by dividing the image into square regions while calculating the mean, minimum and maximum pixel values in each of those regions. The contrast level of the region is decided by the maximum and minimum pixel values which help in

approximating the contrast variation in the output image. Whereas, the error in the output image can be determined by the mean pixel value of the corresponding region.

We need to calculate structural distortions  $S_r$ ,  $S_g$ ,  $S_b$  for the each color components separately. The following formula calculates structural distortion for the red component.

$$S_r = \frac{1}{N} \sum_{x=1}^N \{0.5[Ia_x - Ca_x]^2 + 0.25[Ip_x - Cp_x]^2 + 0.25[Ib_x - Cb_x]^2\} \quad (3.21)$$

Here, mean, minimum and maximum pixel values of the two images selected for the process are indicated by  $Ia_x$ ,  $Ib_x$ ,  $Ip_x$  and  $Ca_x$ ,  $Cb_x$ ,  $Cp_x$ .

Similarly, structural similarities for green and blue components can be calculated.

The overall structural distortion can be calculated as,

$$PSNR_{sd} = 10 \log_{10} \left[ \frac{3}{S_g + S_r + S_b} \right] \quad (3.22)$$

### c. Edge Distortion measurement

Edges tend to strongly impact the image quality. Edges are the areas which display a sudden variation in the pixel value. In the case of HVS, a higher similarity in the edges of the original reference image and distorted image gives higher perceptual quality.

The edge distortion is calculated for each color component individually.

Edge distortion in the red component is given as,

$$ED_r = \frac{1}{MN} \sum_{x=1}^M \sum_{y=1}^N [i_e(x, y) - c_e(x, y)]^2 \quad (3.23)$$

Here,  $i_e(x, y)$  and  $c_e(x, y)$  denote the edge maps of the original and compressed image respectively.

Similarly, edge distortion for the other two components can be calculated denoted by  $ED_g$  and  $ED_b$ .

The overall edge distortion can be calculated as,

$$PSNR_{ed} = 10\log_{10}\left[\frac{3}{ED_g + ED_r + ED_b}\right] \quad (3.24)$$

#### d. Proposed Modified PSNR

It can be referred to as a refined version of the conventional quality metric PSNR and is denoted by  $PSNR_q$  (dB). It is calculated as the weighted sum of the components calculated in previous sections.

$$PSNR_q = 0.32PSNR_{es} + 0.38PSNR_{ed} + 0.3PSNR_{sd} \quad (3.25)$$

The coefficients in the above formula indicate the comparative effect of each distortion. Human eyes notice edge deformation in an image, and hence edge distortion PSNR has a higher weight coefficient. Whereas, error sensitivity and structural distortions affect the nature of an image in a similar way and hence there are assigned approximately equal coefficients.

#### 3.3.5. Structural similarity index

The conventional perceptual image quality evaluation methods adopt a bottom-up approach that conveniently uses the psychophysical features of the HVS, but it is important to note they have their limitations [15].

Structural similarity is a different top-down solution to the issue of assessment of the quality of an image. It assumes that HVS considers the structural information of an image of utmost

importance, and hence the measured structural similarity should provide an acceptable approximation of the image [15].

#### a. Single-Scale Structural Similarity

Consider two images that need to be compared, two image regions are selected. Let  $i = \{i_x \mid x = 1, 2, \dots, N\}$  and  $o = \{o_x \mid x = 1, 2, \dots, N\}$  be two discrete signals. The parameters luminance, contrast, and structure are calculated as,

$$l(i, o) = \frac{2\mu_i\mu_o + C_1}{\mu_i^2 + \mu_o^2 + C_1} \quad (3.26)$$

$$c(i, o) = \frac{2\sigma_i\sigma_o + C_2}{\sigma_i^2 + \sigma_o^2 + C_2} \quad (3.27)$$

$$s(i, o) = \frac{\sigma_{io} + C_3}{\sigma_i\sigma_o + C_3} \quad (3.28)$$

Here,  $\mu_i$ ,  $\sigma_i^2$  and  $\sigma_{io}$  denotes the mean of  $i$ , the variance of  $i$ , and the covariance of  $i$  and  $o$ , respectively.  $\mu_i$  and  $\sigma_i$  indicate the estimates of the luminance and contrast components of  $i$ , and  $\sigma_{io}$  is the tendency of  $i$  and  $o$  to vary together, which indicates structural similarity.  $C_1$ ,  $C_2$ , and  $C_3$  denote constant values expressed as,

$$C_1 = (S_1R)^2, C_2 = (S_2R)^2 \text{ and } C_3 = C_2/2 \quad (3.29)$$

Where,  $R$  denotes the dynamic range of the pixel values,  $S_1$ , and  $S_2$  are scalar constants that are lesser than 1.

The traditional formula to calculate Structural similarity index between the two images  $i$  and  $c$  can be expressed as,

$$SSIM(i,o) = [l(i,o)]^\alpha \cdot [c(i,o)]^\beta \cdot [s(i,o)]^\gamma \quad (3.30)$$

Here,  $\alpha$ ,  $\beta$ , and  $\gamma$  denotes the parameters that indicate the weighted importance of all the color components. In a case where,  $\alpha = \beta = \gamma = 1$ ,

$$SSIM(i,o) = \frac{(2\mu_i\mu_c + C_1)(2\sigma_{ic} + C_2)}{(\mu_i^2 + \mu_c^2 + C_1)(\sigma_i^2 + \sigma_c^2 + C_2)} \quad (3.31)$$

The SSIM algorithm uses Sliding window perspective and moves the window pixel-by-pixel over the region for assessing the image quality. The SSIM value should be in the range 0 and 1 where 1 denotes identical images. As this value decreases from 1 the distinction between the images increases.

#### b. Multi-scale SSIM index

It is preferred for comparing different lossy image compression algorithms. It is applied to each of the RGB channels of an image independently and then averages the results. They give a score between -1 and 1 [14]. This multi-scale outlook provides more flexibility when compared to the single-scale index by integrating the changes in image resolution and conditions like viewing field [15].

In a multi-scale approach, the original input image and the compressed image are given as inputs, which are then passed through a low pass filter and a downsampling stage by a factor of 2. The input image is an index of 1 to N scale range, computed after N – 1 iteration [15].

The overall MS-SSIM is calculated by adding up the measured values at varying scales by the following formula,

$$MS-SSIM(i,o) = [l_N(i,o)]^{\alpha N} \prod_{j=1}^N [c_j(i,o)]^{\beta_j} [s_j(i,o)]^{\gamma_j} \quad (3.32)$$



Here, the  $l_N(i,o)$  denotes the luminance parameter as obtained in (3.26) and is computed only at scale  $N$ ,  $c_j(i,o)$  and  $s_j(i,o)$  denote the contrast parameter obtained in (3.28) and structure parameter obtained in (3.27) at  $j$ th scale respectively. Like (3.30), the exponents  $\alpha_N$ ,  $\beta_j$ , and  $\gamma_j$  are the parameters that are used to adjust the weight coefficients of various. This (3.32) also satisfies the conditions like symmetry and boundedness. As discussed in the previous section, Multi-scale SSIM index must be between -1 and 1. Here, Value of 1 denotes identical images and as it decreases from 1 the distinction between the two images increases.

## CHAPTER IV

### RESULTS AND DISCUSSIONS

In this research, various kinds of images are compressed and analyzed. The images used as an input in a 256x256 height to width ratio. These images are analyzed using the four lossy image compression techniques discussed in chapter 3.

**Wavelet Analysis:** The images will be analyzed using bior4.4 and sym4 wavelets at level 3 decomposition using discrete wavelet transform as discussed in section 3.2.1. These wavelets possess both symmetry and compact support which eliminates the blocking artifacts problem possessed by other wavelets.

**Neural Network Analysis:** The input images are analyzed using a Feed-Forward Neural Network. A network with 16 input layer neurons and 12 hidden layer neurons is simulated at 10 epochs using a Levenberg-Marquardt backpropagation. The images are scanned and then divided into blocks of pixels and passed as vectors to the input layer. The network is then initialized by setting the input layer neurons and hidden layer neurons. The analysis is carried out at 10 epochs to minimize the error and maintain the computational time to a certain acceptable level as discussed in section 3.2.2.

**Hybrid Technique Analysis:** The hybrid techniques will be analyzed using the same input parameters used in Wavelet analysis and Neural Network Analysis as discussed in section 3.2.3 and section 3.2.4.

**Software:** MATLAB software was used for this research analysis because of its simple yet extensive computing environment.

#### 4.1. Result Analysis A

This section aims at comparing the compression results of the input images based on level 3 bior4.4 wavelet decomposition in terms of the performance parameters discussed in section 3.2.5.

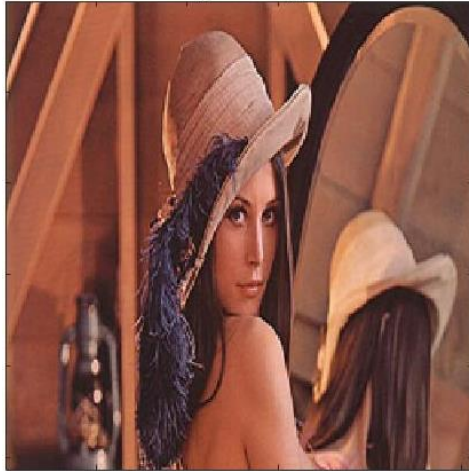
Case 1A - Input Image 1 Analysis using bior4.4 wavelet

Parameters	Wavelet	Neural Network	Hybrid 1 (Wavelet Transform – Neural Network)	Hybrid 2 (Neural Network – Wavelet Transform)
CR	7.4303	7.3147	7.6544	7.5439
PSNR	51.1201	41.0193	40.2630	40.8028
PSNR <sub>es</sub>	55.8913	44.4374	40.3960	41.7099
PSNR <sub>ed</sub>	39.0890	32.9557	33.1088	31.6540
PSNR <sub>sd</sub>	37.6733	36.1130	35.2524	35.6980
PSNR <sub>q</sub>	44.0410	37.9570	36.6638	37.0451
SSIM	0.9790	0.9777	0.9138	0.9423
MS-SSIM				
Level = 3	0.8576	0.8724	0.8200	0.8296
Level = 5	0.8568	0.8641	0.8151	0.8199
Run Time (Sec.)	11.184541	26.154645	40.465622	42.154842

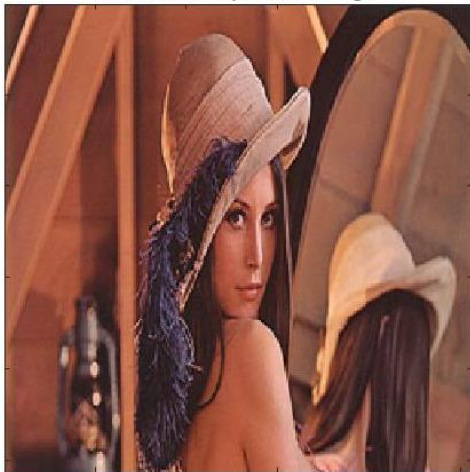
Table 4.1. Parameter Analysis of Case 1A

Observations: As per Table 4.1 and Fig. 4.1, Hybrid 1 technique produces a higher compression ratio in comparison to the other three techniques while maintaining the structural similarity and visual quality of the analyzed images.

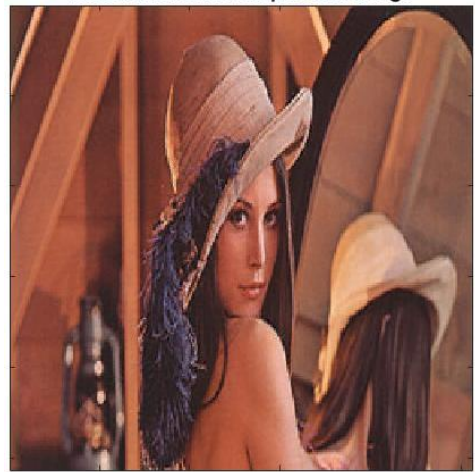
Original Image



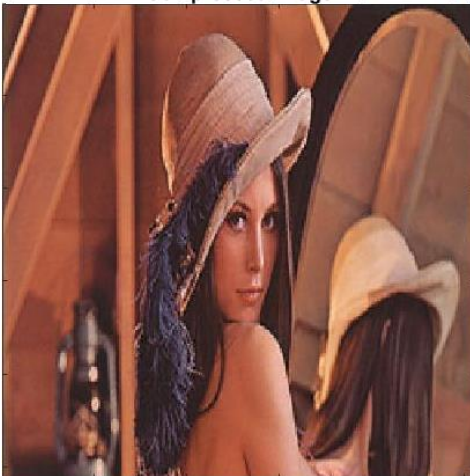
Wavelet Compressed Image



Neural Network Compressed Image



Hybrid 1 (Wavelet Transform - Neural Network)  
Compressed Image



Hybrid 2 (Neural Network - Wavelet Transform)  
Compressed Image

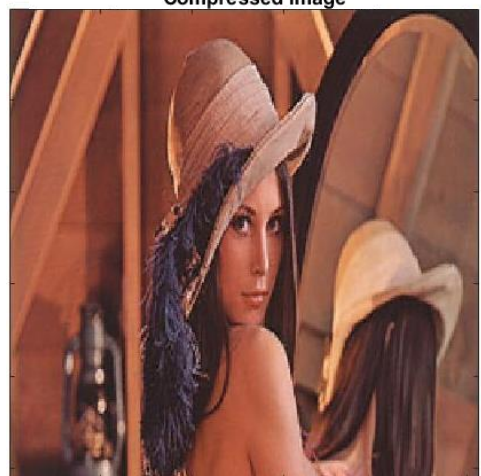


Fig. 4.1. Image Results of Case 1A [19]

Case 2A - Input Image 2 Analysis using bior4.4 wavelet

Parameters	Wavelet	Neural Network	Hybrid 1 (Wavelet Transform – Neural Network)	Hybrid 2 (Neural Network – Wavelet Transform)
CR	7.1717	7.1103	7.6454	7.9735
PSNR	54.1928	36.7743	37.3985	37.4037
PSNR <sub>es</sub>	58.9640	37.8873	38.4616	38.5831
PSNR <sub>ed</sub>	45.3018	33.6275	33.4939	33.4094
PSNR <sub>sd</sub>	23.2935	23.5609	23.5180	23.7537
PSNR <sub>q</sub>	43.0712	33.1707	32.2908	32.7483
SSIM	0.9193	0.9108	0.9029	0.9021
MS-SSIM				
Level = 3	0.7235	0.7152	0.7020	0.7192
Level = 5	0.7394	0.7015	0.7100	0.7255
Run Time (Sec.)	12.256456	24.154452	39.124527	43.254668

Table 4.2. Parameter Comparison of Case 2A

Observations: As per Table 4.2 and Fig. 4.2, Hybrid 2 technique produces a higher compression ratio in comparison to the other three techniques and maintains the structural similarity and visual quality of the analyzed images.

Original Image



Wavelet Compressed Image



Neural Network Compressed Image



Hybrid 1 (Wavelet Transform - Neural Network)  
Compressed Image



Hybrid 2 (Neural Network - Wavelet Transform)  
Compressed Image



Fig. 4.2. Image Results for Case 2A [20]

Case 3A - Input Image 3 Analysis using bior4.4 wavelet

Parameters	Wavelet	Neural Network	Hybrid 1 (Wavelet Transform – Neural Network)	Hybrid 2 (Neural Network – Wavelet Transform)
CR	4.1290	5.9608	6.6521	6.7420
PSNR	57.5192	33.8865	40.3574	40.2034
PSNR <sub>es</sub>	62.2905	40.6578	40.0203	40.5571
PSNR <sub>ed</sub>	47.2286	34.4610	33.5214	33.7347
PSNR <sub>sd</sub>	33.2233	31.3286	31.9415	32.1996
PSNR <sub>q</sub>	47.8468	34.8643	34.4671	33.7973
SSIM	0.9801	0.9049	0.9195	0.9041
MS-SSIM				
Level = 3	0.7831	0.7652	0.7792	0.7612
Level = 5	0.7560	0.7422	0.6905	0.7111
Run Time (Sec.)	12.445783	25.124571	36.247896	39.126853

Table 4.3. Parameter Comparison of Case 3A

Observations: As per Table 4.3 and Fig. 4.3, Hybrid 2 technique produces a higher compression ratio in comparison to the other three techniques and maintains the structural similarity and visual quality of the analyzed images.



Original Image



Wavelet Compressed Image



Neural Network Compressed Image



Hybrid 1 (Wavelet Transform - Neural Network)  
Compressed Image



Hybrid 2 (Neural Network - Wavelet Transform)  
Compressed Image



Fig 4.3. Image Results of Case 3A [21]



Case 4A - Input Image 4 Analysis using bior4.4 wavelet

Parameters	Wavelet	Neural Network	Hybrid 1 (Wavelet Transform – Neural Network)	Hybrid 2 (Neural Network – Wavelet Transform)
CR	2.7939	5.5912	6.9137	6.3236
PSNR	52.6030	30.5602	40.0626	42.7240
PSNR <sub>es</sub>	57.3742	34.3314	34.3071	39.7240
PSNR <sub>ed</sub>	42.2038	29.6883	33.7016	33.8462
PSNR <sub>sd</sub>	24.7859	24.7404	25.0506	24.9969
PSNR <sub>q</sub>	41.8330	30.6897	33.5001	33.4123
SSIM	0.9316	0.9076	0.9029	0.9055
MS-SSIM				
Level = 3	0.7123	0.7030	0.7025	0.7015
Level = 5	0.7012	0.7017	0.7001	0.6974
Run Time (Sec.)	11.853644	22.459812	39.124789	38.178244

Table 4.4. Parameter Comparison of Case 4A

Observations: As per Table 4.4 and Fig. 4.4, Hybrid 1 technique produces a higher compression ratio in comparison to the other three techniques and maintains the structural similarity and visual quality of the analyzed images.

Original Image



Wavelet Compressed Image



Neural Network Compressed Image



Hybrid 1 (Wavelet Transform - Neural Network)  
Compressed Image



Hybrid 2 (Neural Network - Wavelet Transform)  
Compressed Image



Fig. 4.4 Image Results of Case 4A [22]

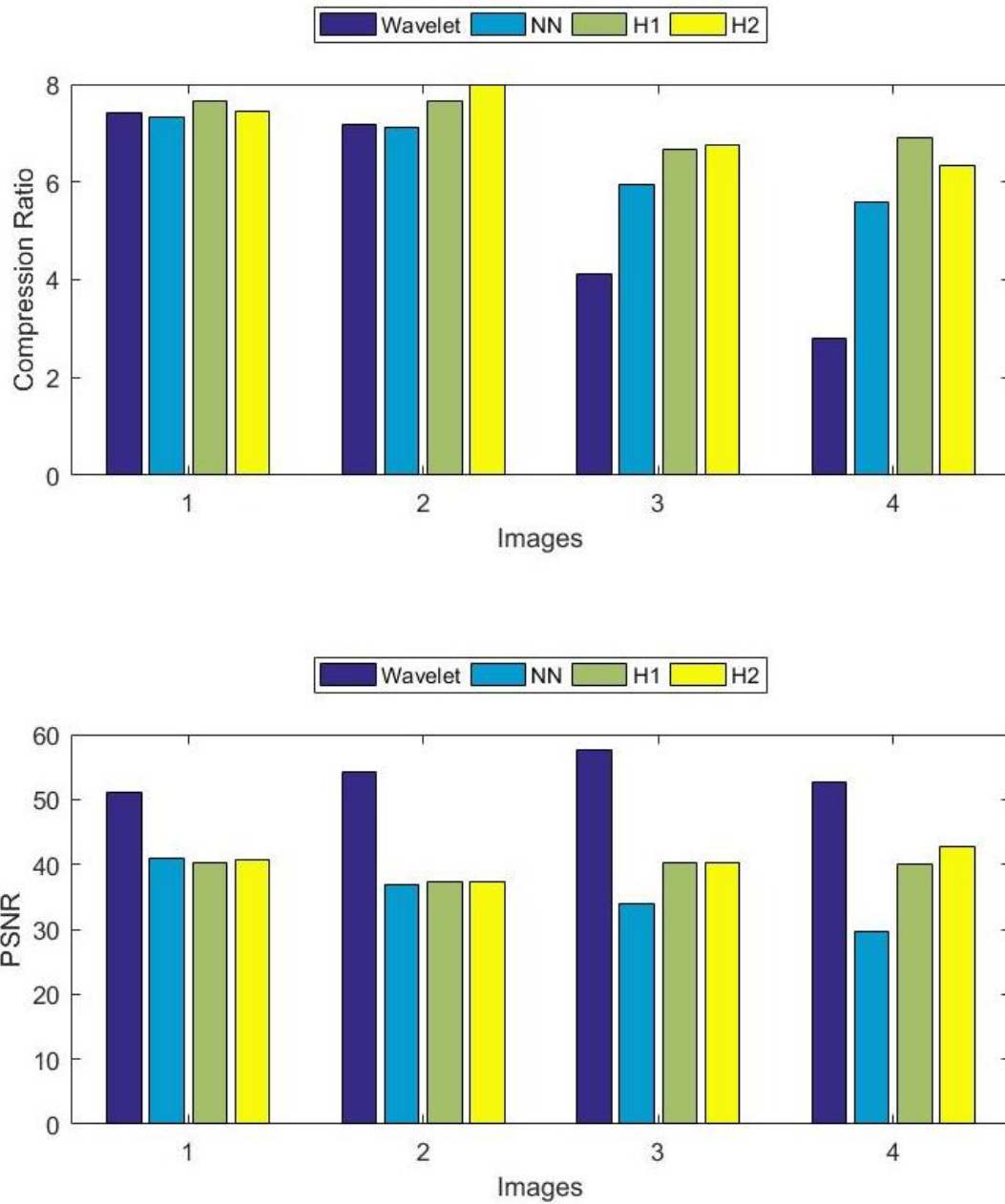


Fig. 4.5. Plot 1 for Comparison of all input images

Observations: Fig. 4.5 compares how all the four images perform in terms of Compression ratio and Peak Signal-to-noise ratio when analyzed using the four image compression techniques.

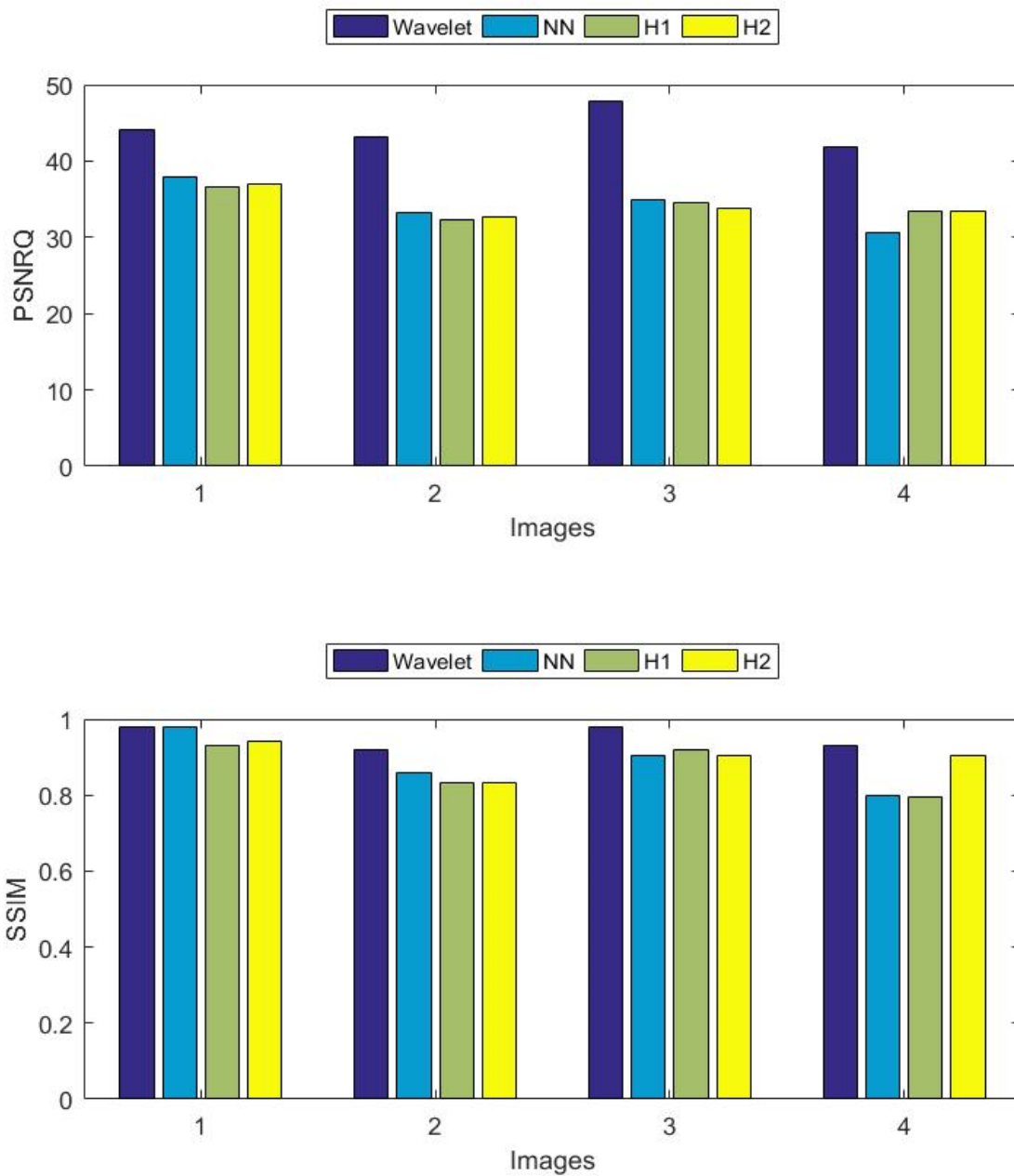


Fig. 4.6. Plot 2 for Comparison of input images

Observations: Fig. 4.6 compares how all the four images perform in terms of Quality Peak Signal-to-noise ratio and Structural Similarity Index when analyzed using the four image compression techniques.

## 4.2. Result Analysis B

This section aims at comparing the compression results of the input images based on level 3 sym4 wavelet decomposition in terms of the performance parameters discussed earlier.

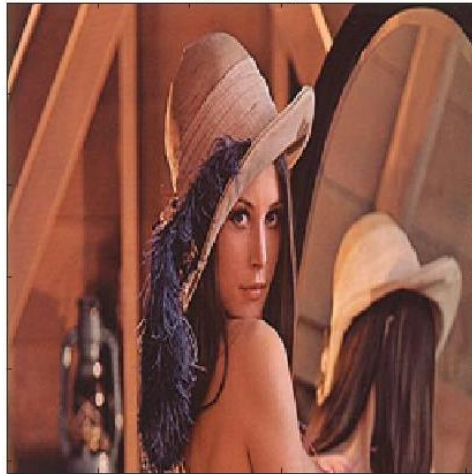
Case 1B- Input Image 1 Analysis using sym4 wavelet

Parameters	Wavelet	Neural Network	Hybrid 1 (Wavelet Transform – Neural Network)	Hybrid 2 (Neural Network – Wavelet Transform)
CR	7.4173	7.3147	7.5478	7.5260
PSNR	51.4334	41.0193	40.2035	41.0797
PSNR <sub>es</sub>	56.2046	44.4374	40.1857	41.2850
PSNR <sub>ed</sub>	39.6594	32.9557	31.3803	31.9530
PSNR <sub>sd</sub>	37.7202	36.1130	35.1866	35.5743
PSNR <sub>q</sub>	44.3721	37.9570	36.2999	36.9856
SSIM	0.9802	0.9777	0.9315	0.9385
MS-SSIM				
Level = 3	0.8587	0.8724	0.8155	0.8201
Level = 5	0.8579	0.8641	0.8078	0.8156
Run Time (Sec.)	11.895612	26.154645	41.198524	42.128536

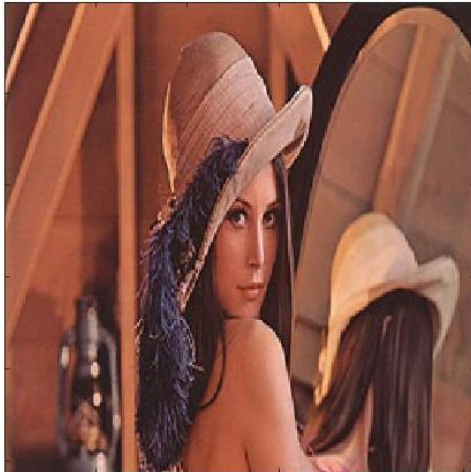
Table 4.5. Parameter Analysis of Case 1B

Observations: As per Table 4.5 and Fig. 4.7, Hybrid 1 technique produces a higher compression ratio in comparison to the other three techniques and maintains the structural similarity and visual quality of the analyzed images.

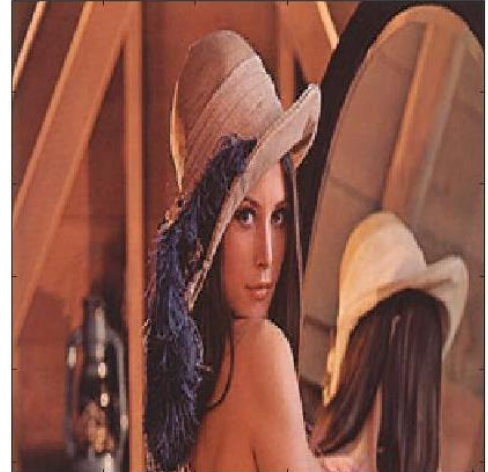
**Original Image**



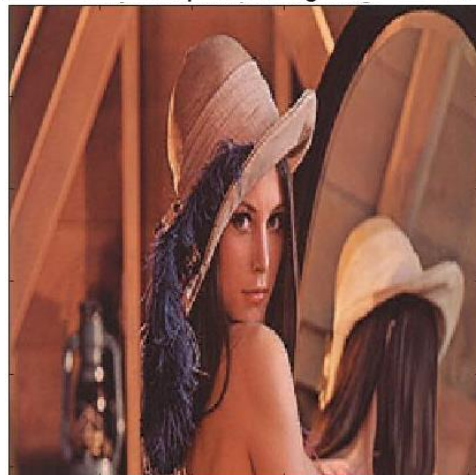
**Wavelet Compressed Image**



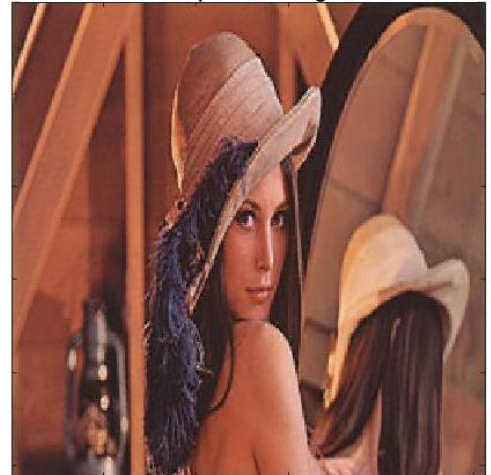
**Neural Network Compressed Image**



**Hybrid 1 (Wavelet Transform - Neural Network)  
Compressed Image**



**Hybrid 2 (Neural Network - Wavelet Transform)  
Compressed Image**



**Fig. 4.7. Image Results of Case 1B [19]**

Case 2B - Input Image 2 Analysis using sym4 wavelet

Parameters	Wavelet	Neural Network	Hybrid 1 (Wavelet Transform – Neural Network)	Hybrid 2 (Neural Network – Wavelet Transform)
CR	7.0995	7.1103	7.5116	7.5272
PSNR	54.4114	36.7743	37.6690	37.5791
PSNR <sub>es</sub>	59.1826	37.8873	38.0762	38.4013
PSNR <sub>ed</sub>	46.1241	33.6275	31.0705	31.0652
PSNR <sub>sd</sub>	23.3043	23.5609	23.6988	23.6100
PSNR <sub>q</sub>	43.4569	33.1707	33.0608	32.1362
SSIM	0.9194	0.8578	0.8029	0.8315
MS-SSIM				
Level = 3	0.7374	0.7152	0.7036	0.6925
Level = 5	0.7402	0.7015	0.6813	0.6801
Run Time (Seconds)	12.965423	24.154452	40.985412	37.985142

Table 4.6. Parameter Comparison of Case 2B

Observations: As per Table 4.6 and Fig. 4.8, Hybrid 2 technique produces a higher compression ratio in comparison to the other three techniques and maintains the structural similarity and visual quality of the analyzed images.



Original Image



Wavelet Compressed Image



Neural Network Compressed Image



Hybrid 1 (Wavelet Transform - Neural Network)  
Compressed Image



Hybrid 2 (Neural Network - Wavelet Transform)  
Compressed Image



Fig. 4.8. Image Results for Case 2B [20]



Case 3B - Input Image 3 Analysis using sym4 wavelet

Parameters	Wavelet	Neural Network	Hybrid 1 (Wavelet Transform – Neural Network)	Hybrid 2 (Neural Network – Wavelet Transform)
CR	4.0793	5.9608	6.3410	6.8149
PSNR	57.7984	33.8865	39.9175	39.6419
PSNR <sub>es</sub>	62.5696	40.6578	38.9718	39.8022
PSNR <sub>ed</sub>	50.4135	34.4610	34.0930	25.9164
PSNR <sub>sd</sub>	33.2318	31.3286	31.7462	31.8316
PSNR <sub>q</sub>	49.1489	34.8643	33.9102	32.1344
SSIM	0.9809	0.9049	0.9140	0.9126
MS-SSIM				
Level = 3	0.7831	0.7652	0.7610	0.7531
Level = 5	0.7560	0.7422	0.7555	0.7492
Run Time (Sec.)	13.226451	25.124571	38.127985	41.235675

Table 4.7. Parameter Comparison of Case 3B

Observations: As per Table 4.7 and Fig. 4.9, Hybrid 2 technique produces a higher compression ratio in comparison to the other three techniques and maintains the structural similarity and visual quality of the analyzed images.

Original Image



Wavelet Compressed Image



Neural Network Compressed Image



Hybrid 1 (Wavelet Transform - Neural Network)  
Compressed Image



Hybrid 2 (Neural Network - Wavelet Transform)  
Compressed Image



Fig 4.9. Image Results of Case 3B [21]

Case 4B - Input Image 4 Analysis using sym4 wavelet

Parameters	Wavelet	Neural Network	Hybrid 1 (Wavelet Transform – Neural Network)	Hybrid 2 (Neural Network – Wavelet Transform)
CR	2.7788	5.5912	7.8233	6.9137
PSNR	52.9744	30.5602	39.1423	40.0222
PSNR <sub>es</sub>	57.7456	34.3314	34.1661	33.9864
PSNR <sub>ed</sub>	43.9868	29.6883	31.1591	31.7811
PSNR <sub>sd</sub>	24.8089	24.7404	24.8806	25.0454
PSNR <sub>q</sub>	42.6363	30.6897	30.1978	30.4261
SSIM	0.9310	0.7976	0.7742	0.8358
MS-SSIM				
Level = 3	0.6917	0.6230	0.6111	0.6190
Level = 5	0.6657	0.6017	0.5989	0.6052
Run Time (Sec.)	13.124581	22.459812	40.356854	43.268255

Table 4.8. Parameter Comparison of Case 4B

Observations: As per Table 4.8 and Fig. 4.10, Hybrid 1 technique produces a higher compression ratio in comparison to the other three techniques and maintains the structural similarity and visual quality of the analyzed images.



Fig. 4.10. Image Results of Case 4B [22]

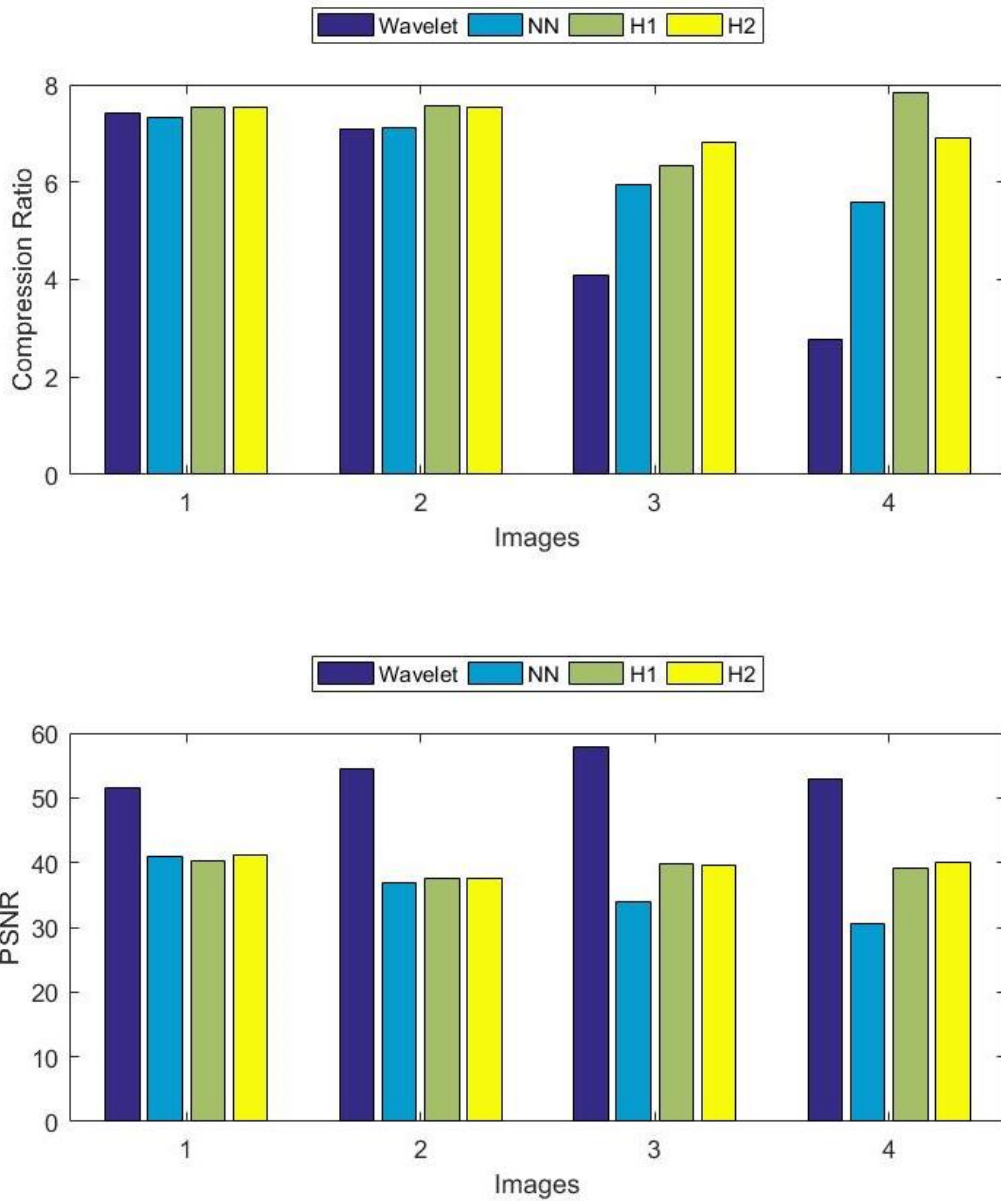


Fig. 4.11. Plot 3 for Comparison of all input images

Observations: Fig. 4.11 compares how all the four images perform in terms of Compression ratio and Peak Signal-to-noise ratio when analyzed using the four image compression techniques.

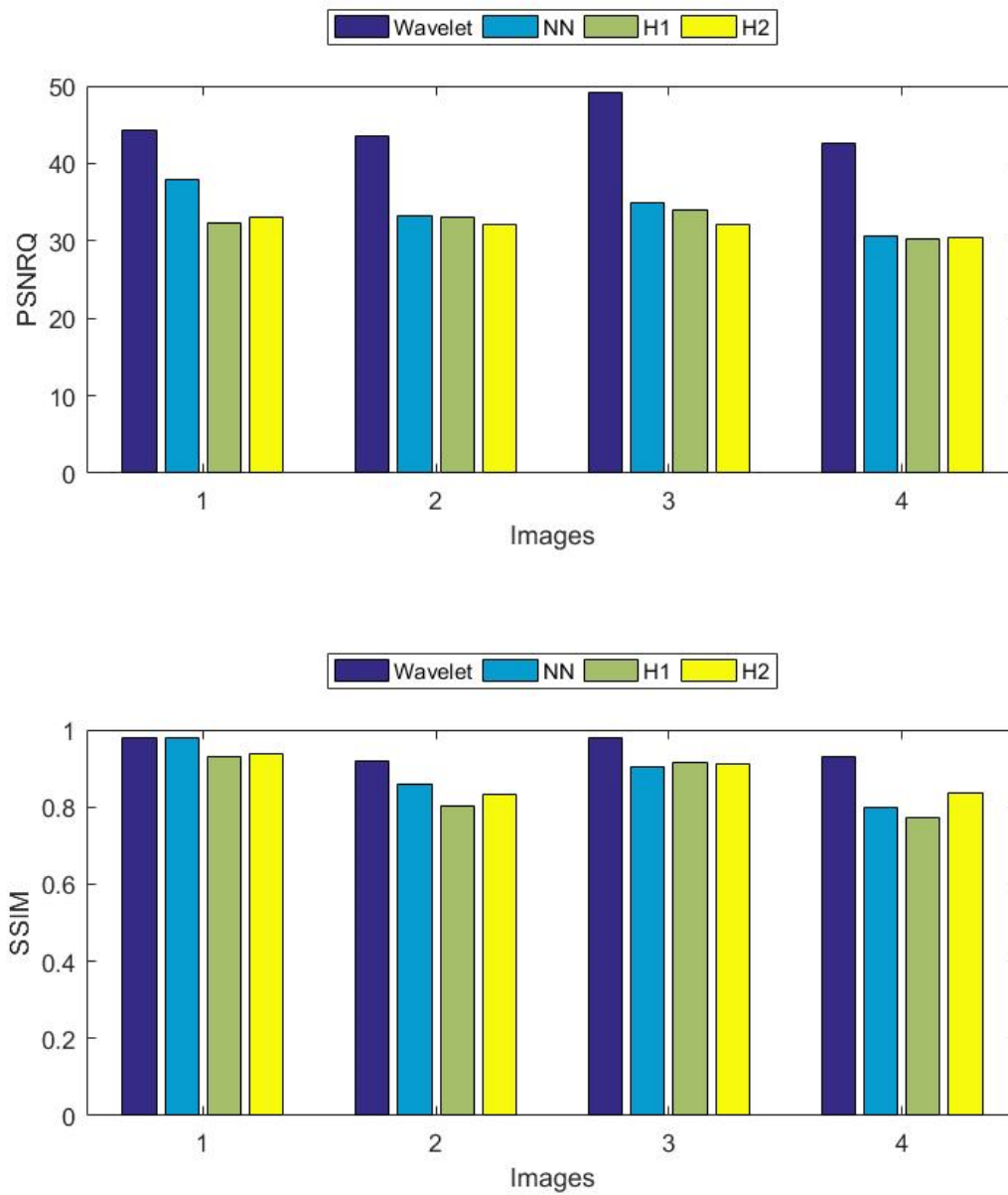


Fig. 4.12. Plot 4 for Comparison of all input images

Observations: Fig. 4.12 compares how all the four images perform in terms of Quality Peak Signal-to-noise ratio and Structural Similarity Index when analyzed using the four image compression techniques.

### 4.3. Result Analysis C

Case 1C – This section aims at comparing the compression results of an input image of size 512x512 based on level 3 bior4.4 wavelet decomposition in terms of the performance parameters discussed earlier. The number of the input layer and hidden layer neurons will be similar as in previous cases, but the epochs will be increased to 100.

Parameters	Wavelet	Neural Network	Hybrid 1 (Wavelet Transform – Neural Network)	Hybrid 2 (Neural Network – Wavelet Transform)
CR	2.2884	5.9842	7.2051	7.1032
PSNR	57.6867	45.4336	50.2299	51.2346
PSNR <sub>es</sub>	57.4579	45.2048	50.0245	50.6153
PSNR <sub>ed</sub>	40.9435	37.7972	38.6776	38.2946
PSNR <sub>sd</sub>	21.2303	21.3577	21.2831	21.3960
PSNR <sub>q</sub>	40.9435	36.5158	37.9314	37.4476
SSIM	0.9765	0.9786	0.9413	0.9680
MS-SSIM				
Level = 3	0.8051	0.7247	0.7716	0.8010
Level = 5	0.7967	0.7103	0.7510	0.7930
Run Time (Sec.)	15.215463	50.135651	145.215348	147.234621

Table 4.9. Parameter Comparison of Case 1C

Observations: As per Table 4.9 and Fig. 4.13, Hybrid 1 technique produces a higher compression ratio in comparison to the other three techniques and maintains the structural similarity and visual quality of the analyzed images. However, higher epochs result in greater Run Time.



Original Image



Wavelet Compressed Image



Neural Network Compressed Image



Hybrid 1 (Wavelet Transform - Neural Network)  
Compressed Image



Hybrid 2 (Neural Network - Wavelet Transform)  
Compressed Image



Fig. 4.13. Image Results of Case 1C [22]



## CHAPTER V

### CONCLUSION

Wavelet-based compression produces higher compression performance by avoiding blocking artifacts as this algorithm does not divide the input image and its essential functions into blocks. Neural Networks based Image compression takes up little computational time as they possess the ability to run in parallel over multiple CPUs. Analysis of Two different hybrid methods combining these two techniques in either order is carried out in this work in terms of the discussed objective and subjective performance parameters. These parameters examine the images in terms of measured values and based on human perception.

Based on the performed analysis, it is noticed that the hybrid methods perform better than the two conventional techniques based on wavelets and neural networks on a case-by-case basis producing higher compression ratios while maintaining the Structural Similarity of the input images and other parameters to acceptable levels. This allows us to send smaller size images in lesser time. The drawback that the hybrid methods possess is comparatively higher computation time and lower peak signal-to-noise ratios, but this is justified as the hybrid methods result in increasing the data storage capacity of the corresponding device and reducing the transmission bandwidth required for data transfer which is an essential characteristic in today's age of large data sets and the need for fast data transmission.

Also, on observation from a naked eye, the original and compressed images are hardly differentiable which means that it preserves the visual quality of the images even after compression.

## CHAPTER VI

### FUTURE SCOPE

Image Processing is an ever-growing field, and with the advancements in today's digital world, we can expect even higher quality images soon. The current compression methods based on the two discussed conventional techniques will always be in fashion. As per the analysis presented in this work, the hybrid approaches can be the next step in image compression world to compress images at even higher compression ratios while maintaining the other performance parameters of the image. The world is now digging deep into artificial intelligence, and the use of neural networks can be highly developed in coming years and contribute towards an innovative technology like the hybrid methods discussed in this research to produce better image compression results.

## REFERENCES

- [1] Khalid Sayood, “Introduction to Data Compression”, Third Edition, ISBN 13: 978-0-12-620862-7, ISBN 10: 0-12-620862-X.
- [2] Ming-yang & Nikolaos Bourbakis, “An Overview of Lossless Digital Image Compression Techniques”, IEEE 48<sup>th</sup> Midwest Symposium on Circuits & Systems, Vol. 2, pp. 1099-1102, Aug. 2005.
- [3] Sumit Chhikara, Reena Sachdeva, “Image Compression using Neural Networks”, Imperial Journal of Interdisciplinary Research, Vol. 2, Issue 07, pp. 1500-1504, 2016.
- [4] Ricardo L. de Queiroz, “Processing JPEG-Compressed Images and Documents”, IEEE Transactions on Image Processing, Vol. 7, No. 12, pp. 1661-1672, December 1998.
- [5] Sunny Arora, Gaurav Kumar, “Review of Image Compression techniques”, International Journal of Recent Research Aspects ISSN: 2349-7688, Vol. 5, Issue 1, March 2018, pp. 185-188.
- [6] Gaurav Kumar, Pradeep Kumar Bhatia, “Empirical Analysis of Image Compression using Wavelets, Discrete Cosine Transform and Neural Network”, 2016 International Conference on Computing for Sustainable Global Development (INDIACom), pp. 3862-3866, 2016.
- [7] A.S. Lewis and G. Knowles, “Image Compression using the 2-D Wavelet Transform”, IEEE Transactions on Image Processing, Vol. 1. No. 2., pp. 244-250 April 1992.
- [8] Alok Kumar Singh, G.S.Tripathi, “A Comparative Analysis of DCT, DWT, & Hybrid (DCT-DWT) Transform Techniques of Image Compression”, IJSRD – International Journal for Scientific Research & Development, Vol. 2, Issue 04, 2014.
- [9] R. Setiono, G. Lu, “Image Compression Using a Feedforward Neural Network”, Proceedings of 1994 IEEE International Conference on Neural Networks (IJCNN), pp. 4761-4765, 1994.

- [10] Vipula Singh, Navin Rajpal, K. Shrikanta Murthy, “Neuro-Wavelet Based Approach for Image Compression”, Computer Graphics Imaging and Visualisation, no. 14, pp. 280-286, Aug. 2007.
- [11] Farhan Hussain, Jechang Jeong, “Exploiting deep neural networks for digital image compression” Web Applications and Networking (WSWAN) 2015 2nd World Symposium on, vol. No, pp. 1-6, 21-23 March 2015.
- [12] Yogita Sawant, L.S. Admuthé, “Hybrid Image Compression Method using ANN and DWT”, International Journal of Computer Applications (0975-8887), Volume 95-No. 1, June 2014.
- [13] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor and Michele Covell, “Full Resolution Image compression using Recurrent Neural Networks” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5306-5314, 2017.
- [14] P. Gupta, P. Srivastava, S. Bhardwaj, V. Bhateja, “A Modified PSNR Metric based on HVS for Quality Assessment of Color Images”, Proceedings of the 2011 International Conference on Communication and Industrial Application (ICCIA 2011), pp. 26-28, Dec. 2011.
- [15] Z. Wang, E.P. Simoncelli, A.C. Bovik, “Multiscale structural similarity for image quality assessment”, Signals Systems and Computers 2004. Conference Record of the Thirty-Seventh Asilomar Conference on, vol. 2, pp. 1398-1402, 2003.
- [16] Sonja Grgic, Mislav Grgic, Branka Zovko-Cihlar, “Performance Analysis of Image Compression using Wavelets”, IEEE Transactions on Industrial Electronics, Vol. 48, No. 3, pp. 682-695, June 2001.
- [17] S. Mishra, S. Savarkar, “Image compression using neural network”, IJCA Proceedings of International Conference and workshop on Emerging Trends in Technology (ICWET 2012) icwet, no. 8, pp. 18-21, 2012.

- [18] K.S. Ng, L.M. Cheng, “Artificial Neural Network for Discrete Cosine Transform and Image Compression”, Proceedings of the Fourth International Conference on Document Analysis and Recognition, vol. 2, pp. 675-678, 1997.
- [19] When the experts in image processing met Miss Lena, ‘21ic Electronic Technology Forum’, 2007. [Online]. Available at: [http://www.ee.cityu.edu.hk/~lmpo/lenna/len\\_top.jpg](http://www.ee.cityu.edu.hk/~lmpo/lenna/len_top.jpg). [Accessed: 23-February-2019].
- [20] Faculty and Staff Resources, ‘Texas A&M University Kingsville’, 2015. [Online]. Available at: <http://www.linkforcounselors.com/wp-content/uploads/2015/09/TAMUK.jpg>. [Accessed: 01- April-2019].
- [21] Goku (@SSJRoseNappa),  
‘<https://twitter.com/SSJRoseNappa/status/1095120596269178880?s=03>’, 11 February 2019.  
Tweet. [Online] Available at: <https://t.co/Z8RVYqQwVB>. [Accessed: 05-March-2019].
- [22] 8 Things That Happen to Your Body When You Don't Eat Enough Fruits and Veggies, ‘Reader’s Digest’, 2016. [Online]. Available at: <https://www.rd.com/wp-content/uploads/2016/09/01-deficient-things-happen-body-fruits-vegetables-AbbieImages.jpg>. [Accessed: 01-April-2019].

## APPENDIX

%Image Compression using Discrete Wavelet Transform

```
tic; %Calculate Run-Time
O=imread(Input Image.jpg'); %Read Input image
X1 = O(:,:,1); X2 = O(:,:,2); X3 = O(:,:,3); %Extract the RGB Channels
[rw cl p] = size(O);
wavelet=input('Select the wavelet ','s'); %Select wavelet for analysis
```

%Level 3 decomposition of the image

```
[c1,l1] = wavedec2(X1,3,wavelet); [c2,l2] = wavedec2(X2,3,wavelet);
[c3,l3] = wavedec2(X3,3,wavelet);
```

%To reconstruct the level 1,2,3 approximation from C and S

```
A31 = wrcoef2('a',c1,l1,wavelet,3); A21 = wrcoef2('a',c1,l1,wavelet,2);
A11 = wrcoef2('a',c1,l1,wavelet,1); H11 = wrcoef2('h',c1,l1,wavelet,1);
V11 = wrcoef2('v',c1,l1,wavelet,1); D11 = wrcoef2('d',c1,l1,wavelet,1);
H21 = wrcoef2('h',c1,l1,wavelet,2); V21 = wrcoef2('v',c1,l1,wavelet,2);
D21 = wrcoef2('d',c1,l1,wavelet,3); H31 = wrcoef2('h',c1,l1,wavelet,3);
V31 = wrcoef2('v',c1,l1,wavelet,3); D31 = wrcoef2('d',c1,l1,wavelet,3);
A32 = wrcoef2('a',c2,l2,wavelet,3); A22 = wrcoef2('a',c2,l2,wavelet,2);
A12 = wrcoef2('a',c2,l2,wavelet,1); H12 = wrcoef2('h',c2,l2,wavelet,1);
V12 = wrcoef2('v',c2,l2,wavelet,1); D12 = wrcoef2('d',c2,l2,wavelet,1);
H22 = wrcoef2('h',c2,l2,wavelet,2); V22 = wrcoef2('v',c2,l2,wavelet,2);
```

```

D22 = wrcoef2('d',c2,l2,wavelet,3); H32 = wrcoef2('h',c2,l2,wavelet,3);
V32 = wrcoef2('v',c2,l2,wavelet,3); D32 = wrcoef2('d',c2,l2,wavelet,3);
A33 = wrcoef2('a',c3,l3,wavelet,3); A23 = wrcoef2('a',c3,l3,wavelet,2);
A13 = wrcoef2('a',c3,l3,wavelet,1); H13 = wrcoef2('h',c3,l3,wavelet,1);
V13 = wrcoef2('v',c3,l3,wavelet,1); D13 = wrcoef2('d',c3,l3,wavelet,1);
H23 = wrcoef2('h',c3,l3,wavelet,2); V23 = wrcoef2('v',c3,l3,wavelet,2);
D23 = wrcoef2('d',c3,l3,wavelet,3); H33 = wrcoef2('h',c3,l3,wavelet,3);
V33 = wrcoef2('v',c3,l3,wavelet,3); D33 = wrcoef2('d',c3,l3,wavelet,3);

```

```

A1(:, :, 1) = A11; A1(:, :, 2) = A12; A1(:, :, 3) = A13;
H1(:, :, 1) = H11; H1(:, :, 2) = H12; H1(:, :, 3) = H13;
V1(:, :, 1) = V11; V1(:, :, 2) = V12; V1(:, :, 3) = V13;
D1(:, :, 1) = D11; D1(:, :, 2) = D12; D1(:, :, 3) = D13;
A2(:, :, 1) = A21; A2(:, :, 2) = A22; A2(:, :, 3) = A23;
H2(:, :, 1) = H21; H2(:, :, 2) = H22; H2(:, :, 3) = H23;
V2(:, :, 1) = V21; V2(:, :, 2) = V22; V2(:, :, 3) = V23;
D2(:, :, 1) = D21; D2(:, :, 2) = D22; D2(:, :, 3) = D23;
A3(:, :, 1) = A31; A3(:, :, 2) = A32; A3(:, :, 3) = A33;
H3(:, :, 1) = H31; H3(:, :, 2) = H32; H3(:, :, 3) = H33;
V3(:, :, 1) = V31; V3(:, :, 2) = V32; V3(:, :, 3) = V33;
D3(:, :, 1) = D31; D3(:, :, 2) = D32; D3(:, :, 3) = D33;

```

```

% 1st level coefficient coding

```

```

a1_cod1 = wcodemat(A11,grayLevels); d1_hcod1 = wcodemat(H11,grayLevels);
d1_vcod1 = wcodemat(V11,grayLevels); d1_dcod1 = wcodemat(D11,grayLevels);
a1_cod2 = wcodemat(A12,grayLevels); d1_hcod2 = wcodemat(H12,grayLevels);
d1_vcod2 = wcodemat(V12,grayLevels); d1_dcod2 = wcodemat(D12,grayLevels);
a1_cod3 = wcodemat(A13,grayLevels); d1_hcod3 = wcodemat(H13,grayLevels);
d1_vcod3 = wcodemat(V13,grayLevels); d1_dcod3 = wcodemat(D13,grayLevels);

```

%2nd level coefficient coding

```

a2_cod1 = wcodemat(A21,grayLevels); d2_hcod1 = wcodemat(H21,grayLevels);
d2_vcod1 = wcodemat(V21,grayLevels); d2_dcod1 = wcodemat(D21,grayLevels);
a2_cod2 = wcodemat(A22,grayLevels); d2_hcod2 = wcodemat(H22,grayLevels);
d2_vcod2 = wcodemat(V22,grayLevels); d2_dcod2 = wcodemat(D22,grayLevels);
a2_cod3 = wcodemat(A23,grayLevels); d2_hcod3 = wcodemat(H23,grayLevels);
d2_vcod3 = wcodemat(V23,grayLevels); d2_dcod3 = wcodemat(D23,grayLevels);

```

% 3rd level coefficients coding

```

a3_cod1 = wcodemat(A31,grayLevels); d3_hcod1 = wcodemat(H31,grayLevels);
d3_vcod1 = wcodemat(V31,grayLevels); d3_dcod1 = wcodemat(D31,grayLevels);
a3_cod2 = wcodemat(A32,grayLevels); d3_hcod2 = wcodemat(H32,grayLevels);
d3_vcod2 = wcodemat(V32,grayLevels); d3_dcod2 = wcodemat(D32,grayLevels);
a3_cod3 = wcodemat(A33,grayLevels); d3_hcod3 = wcodemat(H33,grayLevels);
d3_vcod3 = wcodemat(V33,grayLevels); d3_dcod3 = wcodemat(D33,grayLevels);

```



```

L31 =
[imresize([imresize([a3_cod1,d3_hcod1;d3_vcod1,d3_dcod1],size(d2_hcod1),'bilinear'),d2_hcod
1;d2_vcod1,d2_dcod1],size(d1_hcod1),'bilinear'),d1_hcod1;d1_vcod1,d1_dcod1];

L32 =
[imresize([imresize([a3_cod2,d3_hcod2;d3_vcod2,d3_dcod2],size(d2_hcod2),'bilinear'),d2_hcod
2;d2_vcod2,d2_dcod2],size(d1_hcod2),'bilinear'),d1_hcod2;d1_vcod2,d1_dcod2];

L33 =
[imresize([imresize([a3_cod3,d3_hcod3;d3_vcod3,d3_dcod3],size(d2_hcod3),'bilinear'),d2_hcod
3;d2_vcod3,d2_dcod3],size(d1_hcod3),'bilinear'),d1_hcod3;d1_vcod3,d1_dcod3];

L3(:,:,1) = L31; L3(:,:,2) = L32; L3(:,:,3) = L33;

image(uint8(L3)); axis image; title('Level 3 decomposition');

% Calculate the default parameters and perform the actual compression.

[thres,sorh,nkeep] = ddencmp('cmp','wv',X1);

[Xcomp1,CXC1,LXC1,Perf01,PerfL21]=wdencmp('gbl',c1,l1,wavelet,3,thres,sorh,nkeep);

[thres,sorh,nkeep] = ddencmp('cmp','wv',X2);

[Xcomp2,CXC2,LXC2,Perf02,PerfL22]=wdencmp('gbl',c2,l2,wavelet,3,thres,sorh,nkeep);

[thres,sorh,nkeep] = ddencmp('cmp','wv',X3);

[Xcomp3,CXC3,LXC3,Perf03,PerfL23]= wdencmp('gbl',c3,l3,wavelet,3,thres,sorh,nkeep);

Xcomp(:,:,1) = Xcomp1; Xcomp(:,:,2) = Xcomp2; Xcomp(:,:,3) = Xcomp3;

toc; %Calculate Run-time

%Image Compression using Neural Networks

```

```

r=input('enter input neurons ');           %Set input neurons

h=input('enter hidden layer neurons ');     %Set hidden layer neurons

X1=blkM2vec(X1,[r r]);                     %Block-matrix M to vector count
X2=blkM2vec(X2,[r r]);
X3=blkM2vec(X3,[r r]);

net_c = feedforwardnet(1,'trainlm');        %Initialize the Feed Forward Neural network
net_c.layers{ 1 }.size = h;

net_c.trainparam.epochs=input('enter epochs '); %Set the training parameters
net_c.trainparam.goal=0;

[net_s,tr]=train(net_c,X1,X1);

s=sim(net_s,X1);                           %Simulate the neural networks

C1=vec2blkM(a,r,rw,cl);

s=sim(net_s,X2);

C2=vec2blkM(a,r,rw,cl);

s=sim(net_s,X3);

C3 =vec2blkM(a,r,rw,cl);

Ncomp = cat(3,C1,C2,C3);                    %compute the compressed image


%Calculating Performance Parameters

%To find the compression ratio

in = imfinfo('original.jpg'); disp(in);

out = imfinfo('compressed.jpg'); disp(out);

ib=in.FileSize; disp('The file size of the original image is'); disp(ib);

```

```

cb=out.FileSize; disp('The file size of the Compressed image is'); disp(cb);

cr=ib/cb; disp('The compression ratio is'); disp(cr);

% Calculate mean square error and power signal to noise ratio
mse=(sum(sum(sum((X-Xcomp).*(X-Xcomp)))))/(rw*cl*p);

PSNR=10*log10((255^2)/mse);

disp('Mean square error = '); disp(mse);

disp('PSNR ='); disp(PSNR);

%Calculate Structural Similarity

%Single Scale

ssim_value = ssim(X, RGBImage);

disp('SSIM value is:');

disp(ssim_value(:));

%Multi Scale

ms = msssim(X,Xcomp);

msssim = mean2(ms);

disp('MS-SSIM value is:');

disp(msssim);

%Measurement of Error Sensitivity

I1 = X(:,:,1); I2 = X(:,:,2); I3 = X(:,:,3); %Extract the RGB channels

R1 = Xcomp(:,:,1); R2 = Xcomp(:,:,2); R3 = Xcomp(:,:,3);

D1 = abs(double(I1)-double(R1)).^2;

```

```

Er = (sum(sum(sum(D1(:)))))/(RS1.*RS2.*3);
D2 = abs(double(I2)-double(R2)).^2;
Eg = (sum(sum(sum(D2(:)))))/(RS1.*RS2.*3);
D3 = abs(double(I3)-double(R3)).^2;
Eb = (sum(sum(sum(D3(:)))))/(RS1.*RS2.*3);

PSNRes=10*log10(3/(Er+Eg+Eb));
disp('PSNR error sensitivity is'); disp(PSNRes);

%Measurement of Structural Distortion

%Red Component
Xai = mean(I1(:));
[maxvalXpi,idx]=max(I1, [], 1);
[row,col,pix]=ind2sub(size(I1), idx);
[minvalXbi,idx]=min(I1, [], 1);
[row,col,pix]=ind2sub(size(I1), idx);
Yai = mean(R1(:));
[maxvalYpi,idx]=max(R1, [], 1);
[row,col]=ind2sub(size(R1), idx);
[minvalYbi,idx]=min(R1, [], 1);
[row,col]=ind2sub(size(R1), idx);

i1=0.5*((Xai - Yai)^2)

```

```
i2=0.25*((maxvalXpi - maxvalYpi).^2)
```

```
i3=0.25*((minvalXbi - minvalYbi).^2)
```

```
i = i1 + i2 + i3;
```

```
Sr = sum(i(:))/RS1;
```

```
%Green component
```

```
GXai = mean(I2(:));
```

```
[maxvalGXpi,idx]=max(I2);
```

```
[row,col,pix]=ind2sub(size(I2), idx);
```

```
[minvalGXbi,idx]=min(R2);
```

```
[row,col,pix]=ind2sub(size(R2), idx);
```

```
GYai = mean(R2(:));
```

```
[maxvalGYpi,idx]=max(R2);
```

```
[row,col,pix]=ind2sub(size(R2), idx);
```

```
[minvalGYbi,idx]=min(R3);
```

```
[row,col,pix]=ind2sub(size(R3), idx);
```

```
Gi1=0.5*((GXai - GYai)^2)
```

```
Gi2=0.25*((maxvalGXpi - maxvalGYpi).^2)
```

```
Gi3=0.25*((minvalGXbi - minvalGYbi).^2)
```

```
Gi = Gi1 + Gi2 + Gi3;
```

```
Sg = sum(Gi(:))/RS1;
```

```

%Blue component

BXai = mean(I3(:));

[maxvalBXpi,idx]=max(I3);

[row,col,pix]=ind2sub(size(I3), idx);

[minvalBXbi,idx]=min(I3);

[row,col,pix]=ind2sub(size(I3), idx);

BYai = mean(R3(:));

[maxvalBYpi,idx]=max(R3);

[row,col,pix]=ind2sub(size(R3), idx);

[minvalBYbi,idx]=min(R3);

[row,col,pix]=ind2sub(size(R3), idx);


Bi1=0.5*((BXai - BYai).^2)

Bi2=0.25*((maxvalBXpi - maxvalBYpi).^2)

Bi3=0.25*((minvalBXbi - minvalBYbi).^2)

Bi = Bi1 + Bi2 + Bi3;

Sb = sum(Bi(:))/RS1;


PSNRsd=10*log10(3/(Sr+Sg+Sb));

disp('PSNR structural dist is'); disp(PSNRsd);


% Measurement of Edge distortion

img1=rgb2ycbcr(X);

```

```

dx1=edge(img1(:,:,1),'canny');
dx1=(dx1*255);
img2(:,:,1)=dx1; img2(:,:,2)=img1(:,:,2); img2(:,:,3)=img1(:,:,3);
rslt=ybcr2rgb(uint8(img2));
C=rslt;
C1 = C(:,:,1); C2 = C(:,:,2); C3 = C(:,:,3); %Extract the RGB channels
img11=rgb2ycbcr(RGBImage);
dx11=edge(img11(:,:,1),'canny');
dx11=(dx11*255);
img22(:,:,1)=dx11; img22(:,:,2)=img11(:,:,2); img22(:,:,3)=img11(:,:,3);
rslt1=ybcr2rgb(uint8(img22));
R=rslt1;
C11 = R(:,:,1); C22 = R(:,:,2); C33 = R(:,:,3); %Extract the RGB channels
Dr = abs(double(C1)-double(C11)).^2;
EDr = (sum(sum(sum(Dr(:)))))/(RS1.*RS2.*3);
Dg = abs(double(C2)-double(C22)).^2;
EDg = (sum(sum(sum(Dg(:)))))/(RS1.*RS2.*3);
Db = abs(double(C3)-double(C33)).^2;
EDb = (sum(sum(sum(Db(:)))))/(RS1.*RS2.*3);

PSNRed=10*log10(3/(EDr+EDg+EDb));
disp('PSNR edge distortion is'); disp(PSNRed);

```

%Measurement of Quality PSNR

PSnRq = (0.32\*PSNRes) + (0.38\*PSNRed) + (0.3\*PSNRsd);

disp('Quality PSNR is'); disp(PSnRq);

%Block Matrix to Vector count conversion Function

function vec = blkM2vec(X, blkS)

[h,w,p] = size(X);

r = blkS(1) ; c = blkS(2) ;

if (rem(h, r) ~= 0) || (rem(w, c) ~= 0)

error('blocks do not fit into matrix')

end

x = w/c; y = h/r; N = x\*y; rc = r\*c;

vc = zeros(rc, N, p);

for ii = 0:y - 1

vc(:,(1:x)+ii\*x) = reshape(X((1:r)+ii\*r,:),rc,x);

end

% Vector count to Block Matrix conversion Function

function X = vec2blkM(vec, r, h, w)

%vec2blkM reshapes a matrix vec of rc by 1 vectors into a block-matrix X of xh by ph size

% Each rc-element column of vec is converted into a r by c block of a matrix X and placed as a

block-row element

[VectorsSize ,VectorCount] = size(vec) ;



```

px = VectorsSize*VectorCount ;

if ( (rem(px, h) ~= 0) || (rem(h, r) ~= 0) )

    error('number of rows of the matrix error')

end

if ( (rem(px, w) ~= 0) || (rem(w, r) ~= 0) )

    error('number of rows of the matrix erro')

end

ph = px/h;

if ( (rem(VectorsSize, r) ~= 0) || (rem(VectorCount*r, h) ~= 0) )

error('block size error')

end

c = VectorsSize/r ;

xh = zeros(r, VectorCount*c);

xh(:) = vec ;

nrb = h/r ;

X = zeros(h, ph);

for ii = 0:nrb-1

X((1:r)+ii*r, :) = xh(:, (1:ph)+ii*ph) ;

end

%Structural Similarity Function

function mssim = msssim(X, Xcomp, K, window, scale, weight, method)

K = [0.01 0.03];

```

```

window = fspecial('gaussian', 11, 1.5);

scale = 5;

weight = [0.1448 0.2856 0.2001 0.2363 0.1333];

method = 'product';

[H1 W1] = size(win);

min_img_width = min(H, W)/(2^(scale-1));

max_win_width = max(H1, W1);


im1 = double(X); im2 = double(Xcomp);

C1 = (K(1)*L)^2; C2 = (K(2)*L)^2;

win = window/sum(sum(window));

mu1 = filter2(win, im1, 'valid');

mu2 = filter2(win, im2, 'valid');

mu1_sq = mu1.*mu1;

mu2_sq = mu2.*mu2;

mu1_mu2 = mu1.*mu2;

sigma1_sq = filter2(win, im1.*im1, 'valid') - mu1_sq;

sigma2_sq = filter2(win, im2.*im2, 'valid') - mu2_sq;

sigma12 = filter2(win, im1.*im2, 'valid') - mu1_mu2;

num1 = 2*mu1_mu2 + C1;

num2 = 2*sigma12 + C2;

den1 = mu1_sq + mu2_sq + C1;

den2 = sigma1_sq + sigma2_sq + C2;

```

```

ssim_map = ones(size(mu1));

index = (denominator1.*denominator2 > 0);

ssim_map(index) = (num1(index).*num2(index))./(den1(index).*den2(index));

index = (den1 ~= 0) & (den2 == 0);

ssim_map(index) = num1(index)./den1(index);


ssim = mean2(ssim_map);

downsamplefilter = ones(2)./4;

for l = 1:scale

[mssim_array(l) ssim_map_array{1} mcs_array(l) cs_map_array{1}] = ssim(im1, im2, K,
window);

    filtered_im1 = imfilter(im1, downsamplefilter, 'symmetric', 'same');

    filtered_im2 = imfilter(im2, downsamplefilter, 'symmetric', 'same');

    im1 = filtered_im1(1:2:end, 1:2:end);

    im2 = filtered_im2(1:2:end, 1:2:end);

end

if (method == 'product')

mssim = prod(mcs_array(1:scale-1).^weight(1:scale-1))*(mssim_array(scale).^weight(scale));

else

weight = weight./sum(weight);

mssim = sum(mcs_array(1:scale-1).*weight(1:scale-1)) + mssim_array(scale).*weight(scale);

```

## VITA

Bhavik Prashant Bhoir is currently pursuing a Master of Science degree in Electrical Engineering at Texas A&M University – Kingsville, Texas, USA. He received a bachelor's degree in Electronics and Telecommunications Engineering from University of Mumbai, Mumbai, India in 2017.

He was a graduate teaching assistant with the Department of Electrical Engineering and Computer Science Department at Texas A&M University – Kingsville, Texas, USA during Fall 2018 semester.