An Algorithmic Approach for Efficient Image Compression using Neuro-Wavelet Model and Fuzzy Vector Quantization Technique

Vipula Singh[1*], Navin Rajpal[2], K. Srikanta Murthy[3]

1. Electronics and communication department, Geethanjali College of Engineering Hyderabad India
2. School of Information Technology GGSIP University New Delhi, India.
3. Information science department PESIT BSK III Bangalore 560085 India.

Abstract: Applications, which need to store large database and/or transmit digital images requiring high bit-rates over channels with limited bandwidth, have demanded improved image compression techniques. This paper describes practical and effective image compression system based on neuro-fuzzy model which combines the advantages of fuzzy vector quantization with neural network and wavelet transform. The emphasis here is on the usefulness of fuzzy vector quantization when it is combined with conventional image coding techniques. The implementation consists of three steps. First, the image is decomposed at different scales using wavelet transform to obtain an orthogonal wavelet representation of the image Each band can be subsequently processed in parallel. Thus, the processing speed can be much faster than otherwise. Different quantization and coding schemes are used for different sub bands based on their statistical properties. At the second step, wavelet coefficients corresponding to lowest frequency band are compressed using differential pulse code modulation. Neural network is used to extract the principal components of the higher frequency band wavelet coefficients. Finally, results of the second step are used as input to the fuzzy vector quantization algorithm. Our simulation results show encouraging results and superior reconstructed images are achieved. The effect of noise on the compression performance is also studied.

Keywords: Image Compression, Fuzzy Vector Quantization, Multiresolution Analysis, Neural Network, noise.

1. Introduction

Digital image presentation requires a large amount of data and its transmission over communication channels is time consuming. Numerous lossy image compression techniques have been developed in the past years[2, 5, 6, 11]. The transform based coding techniques, and in particular block-transform coding, have proved to be the most effective in obtaining large compression ratios while retaining good visual quality. Cosine transform based techniques (JPEG) have been found to obtain excellent results in many digital image compression applications [16]. Vector quantization (VQ) offers good performance when high compression rates are needed [9]. In practice, however, the existing VQ algorithms, often, suffer from a number of serious problems, e.g., long search process, codebook initialization, and getting trapped in local minima, inherent to most iterative processes. To eliminate these problems, a multi-resolution codebook is generated using fuzzy clustering techniques. These clustering techniques integrate fuzzy optimization constraints with the fuzzy- c-means algorithm [3, 13]. The resulting multi resolution codebooks generated from the wavelet decomposed images yield significant improvement in the coding process. Noise degrades the performance of any image compression algorithm [21, 22, 23]. Noise can occur during the image capture, transmission or processing and may be dependent on or independent of image content.
The main objective of this paper is to present a Neuro-Fuzzy model based on wavelet transform and study the effect of noise on the image compression algorithm. The paper is organized as follows: Section 2 discusses the compression method in detail. Section 3 reports sample simulation results and section 4 provides concluding remarks.

2. Image compression model

Fig 1 shows the block diagram of the image encoder. The multiresolution nature of discrete wavelet transform is a powerful tool to represent images decomposed along the vertical and horizontal directions using the pyramidal multiresolution scheme. The wavelet transform decomposes the images into a set of sub images with different resolutions corresponding to different frequency bands.

2.1 Discrete Wavelet Transform

In the first step, wavelet transform is used to decompose the image into seven sub-bands. A typical 2-D DWT, used in image compression, generates the hierarchical pyramidal structure. Fig 2 shows a two-level wavelet decomposition scheme for a 512*512 digital image. This decomposition scheme produces three side bands of size 256*256 corresponding to resolution level 1 and three side bands of size 128*128, corresponding to resolution level 2. Sub-band

1 represents the lowest frequency components of the original image. Much of the energy of image is concentrated here. Sub-band 2 to sub-band 7 contains detail information of edge, outline of image at different decomposition layers. Sub- band 3 and 5 denote coefficients of image at vertical edge after the first and second layers wavelet decomposition. Sub-band 2 and 6; denote coefficients of image at horizontal edge. Sub-band 4 and 7, denote the coefficients of image on the cross edge.

2.2 Neural Network for Image Compression

$$u_j(x_i) = \begin{cases} 1 & if\ dis(x_i, c_j) = dis_{\min}(x_i) \\ 0 & otherwise \end{cases} \quad (4)$$

Because the human visual system has different sensitivity to different frequency bands, the following strategy is adapted for image encoding. A predetermined compression ratio is used for each sub image in the pyramidal structure. The lowest frequency band, band 1 (fig 2) is encoded with Differential Pulse Code Modulation (DPCM). After that these coefficients are scalar quantized. The remaining frequency bands, sub-band 2 to 6 are coded using neural network. As sub-band 2 and 3 contain the similar frequency contents for different orientation, both the bands are encoded using the same neural network with eight units input layer nodes, eight output layer nodes and 6 hidden layer nodes (fig 3) i.e 8-6-8 neural network. Sub-band 5 and 6 have the similar frequency contents for different orientation and are coded using 16-1-16 network. Sub-band 4 coefficients are coded using separate 8-4-8 neural networks as frequency characteristics of these bands do not match with other band. Sub-band 7 information is discarded as it contains little information to contribute to the image. From this stand, this band can be assumed to be zero with little effect on the quality of reconstructed image.

If we denote the weight from x_i to h_j as w_{ij}, then the k elements of hidden vector h are related to n elements of input vector x by equation (1).

$$h_i = \sum_{i=1}^{n} w_{ij} x_i \qquad \text{j=1, 2, ……k} \quad (1)$$

The weight between h_j and y_i is w_{ij}, then the n elements of hidden vector h by equation (2).

$$y_i = \sum_{i=1}^{k} w_{ij} h_j \qquad \text{=1, 2, ……n} \quad (2)$$

The weights of neural network are chosen to minimize the distortion between the input vector and output vector. Back propagation algorithm is used to minimize the squared error between input vector and output vector for the training set. $x \in [-1,1]$ denotes the normalized values of the wavelet coefficients. With back propagation neural network, compression is achieved in two phases, training phase and encoding.

To train a single network corresponding to a series of images having similar properties and statistics at first, and then transmit the output of the hidden layer only for a new presented and compressed image. Given that the network is already trained in advance (off-line), the problems of real-time training and convergence are circumvented. After training, the network is ready for operational use. The wavelet coefficients are presented to the network one at a time. The outputs of the hidden layer nodes constitute the compressed features of an input block. To achieve compression in a practical sense, the outputs of the hidden layer nodes are quantized.

2.3. Quantization

The output of the hidden layer of neural network is quantized to achieve further compression. Various vector quantization algorithms like k-means, FVQ1, FVQ2 and FVQ3 were used for code book design in different sets of experiments. Finally these quantized values are entropy encoded. Huffman encoding is used here.

2.3.1 k-means Algorithm

The k-means algorithm partitions a collection of N vector into k clusters C_j, j=1,....,k. The aim of the algorithm is to find cluster centers (centroids) for each group. The algorithm minimizes a dissimilarity (or distance) function which is given in equation (3).

$$D = \frac{1}{N}\sum_{i=1}^{N} dis_{\min}(x_i) = \frac{1}{N}\sum_{i=1}^{N} \min_{j \in C} dis(x_i, c_j) \quad (3)$$

c_j is the centroid of cluster j; $dis(x_i, c_j)$ is the distance between j^{th} centroid c_j and i^{th} data point x_i. For simplicity, the Euclidian distance is used as dissimilarity measure.

Partitioned groups can be defined by a k x n binary membership matrix (u), where the element u_{ij} is 1 if the i_{th} data point x_i belongs to group j and 0 otherwise as shown in equation (4)

Where dis_{min} is the minimum distance between x_i and c_j. Codebook vectors are evaluated by a distortion measure defined in equation (5)

$$J = \sum_{j=1}^{k}\sum_{i=1}^{N} u_j(x_i) \left\| x_i - c_j \right\|^2 \quad (5)$$

Centroids are computed as the mean of all vectors in group j:

$$c_j = \frac{\sum_{i=1}^{N} u_j(x_i)\, x_i}{\sum_{i=1}^{N} u_j(x_i)} \ \forall\ j = 1,2,\ldots,k \quad (6)$$

The k-means algorithm is summarized in table I

Table I
k-means Algorithm

start
1. Select Randomly $C = \{c_j, j = 1,2,\ldots, k\}$
2. Evaluate u using (4).
3. Compute J using (5). Stop if its improvement over previous iteration is below a threshold.
4. Update c_j according to (6). Go to step 2.

stop

While the k-means algorithm converges to a local minimum, it is not guaranteed to reach the global minimum. In addition, the algorithm is very sensitive to the initial codebook. Furthermore, the algorithm is slow since it requires an exhaustive search through the entire codebook during each iteration.

2.3.2 Fuzzy Vector Quantization Algorithms

These algorithms were developed by evolving a fuzzy codebook design process strategy for the transition from soft to crisp decision and defining conditions for constructing a family of membership functions [13]. In the start of the algorithm, the training vectors are assigned to the codebook vectors that are included in a hemisphere centered at the training vector. The membership function tells the possibility of the training vector belonging to a certain cluster. Thus membership function is a decreasing function of distance and takes values between 0 and 1. Let $P_i^{(t)}$ be the set of the codebook vectors belonging to the hemisphere centered at the training vector during the t^{th} iteration. Initially during the start of the clustering process, each training vector is assigned to every cluster i.e. $P_i^{(0)} = C$, where c is a finite set containing k codewords. After the t^{th} iteration, the hemisphere located at training vector x_i the includes the vectors $c_j \in P_i^{(t)}$ whose distance from x_i is less than equal to the average distance between x_i and $c_j \in P_i^{(t)}$ given by

$$dis_{avg}^{(t)}(x_i) = \frac{1}{N(P_i^{(t)})} \sum_{c_j \in P_i^{(t)}} dis(x_i, c_j) \tag{7}$$

where $N(P_i^{(t)})$ is the total number of elements in the set $P_i^{(t)}$. The set $P_i^{(t+1)}$ is formed by

$$P_i^{(t+1)} = \{c_j \in P_i^{(t)} : dis(x_i, c_j) \le dis_{avg}^{(t)}(x_i)\} \tag{8}$$

By gradually reducing the radius of the hemisphere, the transition from fuzzy to crisp mode is achieved during the algorithm. When the codebook vector contains a single element, the training vector is assigned to that cluster with the closest center. i.e. if $N(P_i^{(t)}) < 2$ training set x_i is transferred from fuzzy to crisp mode.

The membership value indicates the extent to which a training vector belongs to a particular cluster. A valid membership function must follow the following properties:

1. If the hemisphere centered at x_i includes a single codebook vector c_{j*}, the x_i is assigned to j^{*th} cluster.
$$u_j(x_i) = 1 \; if \; dis(x_i, c_j) = dis(x_i, c_{j*})$$
$$u_j(x_i) = 0 \; if \; dis(x_i, c_j) \neq dis(x_i, c_{j*})$$
2. If the number of elements in the set $P_i^{(t)}$ is more than one, then the membership function depends on the distance between x_i and $c_j \in P_i^{(t)}$, that is
$$u_j(x_i) = f(dis(x_i, c_j), c_j \in P_i^{(t)})$$

$u_j(x_i)$ must satisfy following requirements
a) $u_j(x_i)$ is a decreasing function of $dis(x_i,c_j)$
b) $u_j(x_i)$ approaches unity as $dis(x_i,c_j)$ approaches zero.
c) $u_j(x_i)$ approaches zero as $dis(x_i,c_j)$ approaches $dis_{max}(x_i)$ where $dis_{\max}(x_i) = \max_{c_j \in P_i^{(t)}} dis(x_i, c_j)$.

Using different membership functions and different methods for codebook calculation, three FVQ algorithms are explained as follows.

a) Fuzzy Vector Quantization 1 (FVQ1)
This algorithm was developed by constructing a family of membership functions satisfying the conditions proposed in the section 2.3.2 [17]. FVQ1 algorithm uses the membership function given in equation (9)

$$u_j(x_i) = f(dis(x_i, c_j), dis_{\max}(x_i)) = \left(1 - \frac{dis(x_i, c_j)}{dis_{\max}(x_i)}\right)^{\mu} \tag{9}$$

where μ is a positive integer. The vector assignment is based on crisp decisions toward the end of the algorithm. This is guaranteed by the minimizing the discrepancy measure J defined in equation (5) with respect to c_j which results in the equation (6) for the computation of c_j. This selection ensures that the algorithm reduces to crisp k-means algorithm after all the training vectors are transferred from fuzzy to crisp mode. The Fuzzy Vector Quantization 1 (FVQ1) algorithm is summarized in table II.

b) Fuzzy Vector Quantization 2 (FVQ2)
This algorithm uses the membership function of equation (10) which satisfies the conditions proposed in the section 2.3.2.

$$u_j(x_i) = \frac{1}{\sum_{l=1}^{k} \left(\frac{dis(x_i, c_j)}{dis(x_i, c_l)} \right)^{\frac{1}{m-1}}} \tag{10}$$

The codebook vectors are evaluated by equation (11), resulting from the minimization of J=J (c_j , j = 1, 2,.. . , k).

$$c_j = \frac{\sum_{i=1}^{N} u_j(x_i)^m \; x_i}{\sum_{i=1}^{N} u_j(x_i)^m} \; \forall j = 1,2,\ldots,k \tag{11}$$

In this algorithm, toward the end of the algorithm, the training vector assignment is entirely based on crisp decision and $u_j(x_i)$ takes the value zero and one, regardless of the value of m, $u_j(x_i)^m = u_j(x_i)$. The Fuzzy Vector Quantization 2 (FVQ 2) algorithm is summarized in table II.

c) Fuzzy Vector Quantization 3 (FVQ3)

This algorithm uses the membership function of equation (10) which satisfies the conditions proposed in section 2.3.2. The codebook vectors are evaluated using crisp formula in equation (6) which guarantees that the codebook vectors are not affected by the training vectors that are assigned a membership value smaller than unity. i.e. if $m \approx 1$, $u_j(x_i) \approx 1$ and $u_j(x_i)^m \approx u_j(x_i)$ and for fixed m, $(u_j(x_i)^m - u_j(x_i))$ increases as $u_j(x_i)$ approaches zero. Another advantage of using equation (6) over the equation (11) is that the former is less computationally demanding. The Fuzzy Vector Quantization 3 (FVQ 3) algorithm is summarized in table II.

Table II
THE FVQ 1, FVQ2, AND FVQ3 ALGORITHMS

start
Select ε and ε' ($\varepsilon' > \varepsilon$)
Select Randomly $C = \{c_j, j = 1,2,\ldots, k\}$
Set $Q = \phi$; $P_i = C \; \forall \; i = 1,2, ..., N$
Evaluate *D* according to (3)
1 $D^* = D$
i=0
2 $i \leftarrow i + 1$
if $x_i \in Q_i$; then:
Evaluate $u_j(x_i)$ based on the nearest neighbor condition
go to 3
else:
if $x_i \notin Q$ and $N(P_i^{(t)}) > 2$; then:
if $c_j \notin P_i$; then: $u_j(x_i) = 0$
if $c_j \in P_i$; then: Evaluate $u_{ij}(x_i)$ using (9)
[FVQl algorithm]
if $c_j \in P_i$; then: Evaluate $u_j(x_i)$ using (10)
[FVQ2 & FVQ3 algorithms]
Evaluate $dis_{avg}(x_i)$ according to (7)
Update the set P_i according to (8)
else:
if $x_i \notin Q$ and $N(P_i^{(t)}) < 2$; then:
$Q \leftarrow Q \cup \{ x_i \}$
Evaluate $u_j(x_i)$ based on the nearest neighbor condition
3 if $i < N$; then: go to 2
Evaluate c_j using (6) [FVQI & FVQ3 algorithms]
Evaluate c_j using (11) [FVQ2 algorithm]
Evaluate *D* according to (3)
if $Q = X$; then: go to 4
if $(D - D^*)/D^* > \varepsilon'$; then: go to 1
$Q = X$
4 if $(D - D^*)/D^* > \varepsilon$; then: go to 1
stop

3. Experimental results and discussion

In this section, we evaluate compression performance of our image encoder and compare its performance with state of the art image encoders like JPEG and JPEG2000. The influence of noise on compression performance is also studied. The image encoder and decoder was implemented in Matlab 7 on an intel Pentium 2.66 GHz machine with 448 MB RAM. Experiments were conducted using different standard test images of size 512 x 512, with $2^8 = 256$ gray levels. A set of experiments evaluated the effect of different wavelet filters on the quality of the reconstructed image. Images were decomposed using Daubechies' 1- coefficient filter (DAUB 1), 6- coefficient filter (DAUB 6) and 18 coefficient filter (DAUB 18), coiflet-5 coefficient filter and biorthogonal-9 coefficient filter [1]. In different sets of experiments, k-means algorithm and FVQ1, FVQ2 and FVQ3 algorithms were used for vector quantization on the coefficients of hidden layer. For FVQ1, the membership function was evaluated using equation (9) with μ ranging from 1 to 5. The membership function for FVQ2 and FVQ3 was evaluated using equation (10). The parameter m was evaluated as $m = 1 + 1/\lambda$, where $\lambda = 1$ to 5. Data independent additive white Gaussian noise with zero mean and variance varying from 0.001 to 0.02 and salt & pepper noise with density varying from 0.01 to 0.2 was added to the standard test images. Noisy input images were presented to the algorithm and effect of noise on the performance was evaluated.

3.1 Image quality evaluation

The image quality can be evaluated objectively and subjectively. A standard objective measure of image quality is reconstruction error. Two of the error metrics used to compare the various image compression techniques are the mean square error (MSE) and the Peak Signal to Noise Ratio (PSNR). The MSE is the cumulative squared error between the compressed and the original image, whereas PSNR is a measure of the peak error. The mathematical formulae for the computation of MSE & PSNR is

$$MSE = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (I_{ij} - I'_{ij})^2 \quad (12)$$

$$\text{PSNR} = 20 * \log10 (255 / \text{sqrt}(\text{MSE})) \quad (13)$$

where I(x,y) is the original image, I'(x,y) is the approximated version (which is actually the decompressed image) and M, N are the dimensions of the images, 255 is the peak signal value. Subjective quality is measured by psychophysical tests and questionnaires.

3.2 Comparative Study

This set of experiments evaluates the performance of various vector quantization algorithms for codebook design

on various test images. Table III presents PSNR(db) and compression(bpp) of different test images compressed by db18 and codebook designed using k-means, FVQ1, FVQ2 and FVQ3 algorithms The image compression system was trained using the image "lena" and different standard test images were tested on it. It is observed that FVQ algorithms for codebook designing resulted in better SNR values with better visual quality of the reconstructed image as compared to k-means algorithm. This can be accounted to that the fuzzy vector quantization algorithms make use of the uncertainty to the benefit of the clustering process. The algorithm is based on a flexible strategy allowing the transition from soft to crisp decisions which eliminates the effect of initial codebook selection on the quality of clustering. FVQ2 and FVQ3 algorithms achieve best codebook design among all the algorithms used here with FVQ3 algorithm resulting in a slightly better codebook than that provided by FVQ2 algorithm. On an average, the performance of FVQ1 algorithm is comparable to that of FVQ2 and FVQ3 algorithms, which are computationally more demanding than the FVQ1 algorithm. FVQ3 algorithm requires slightly less arithmetic operations per iteration as compared to the FVQ2 algorithm. Figure 5 illustrates the results. It is interesting to note that PSNR of the test images 'woman with dark hair' and 'house' is higher than the training image 'Lena'. One possible interpretation for this result is that the network trained on a much complex image Lena which contains richer features that have been represented so effectively that it is capable of producing better results to simpler images. Same is true for more complex images like 'peppers' where the resulting PSNR value is lower than the test image.

3.3 Effect of noise on compression

If an image has been corrupted by additive noise, the idea of using a lossy compression algorithm to denoise the signal is proposed in several works. Ai-Shaykh [21] et al. studied the effect of noise on image compression using the JPEG lossy image processing standard, where it was found that at higher compression rates the coders filtered out most of the noise, but also degraded the image quality measured by Peak Signal to Noise Ratio (PSNR). Cosman et al. [22] noticed in their evaluation of the quality of compressed medical images, that slightly vector-quantized images are often superior to the originals because noise is suppressed by a clustering algorithm. Lo et al [23] discussed the effect of noise on lossless medical image compression. They concluded that noise decreases the compression ratio. This is because the noise reduces inter-pixel correlation.

In most cases, the quality of the decompressed image is closer to the original image than that of input of the coder. When compressing degraded images, we should be concerned with preserving original information in the image not preserving the noise. When the images are noisy, the fidelity criterion should depend on the original image, not on the input of the coder. One main question which arises is how much has noise influence on the compression performance? To study the effect of noise on compression, the 512 x 512 Lena image is corrupted by data independent, additive Gaussian noise and salt and pepper noise. Following three scenarios are considered to test the noise reduction capability of the neural network in the algorithm.

Case I) *Noiseless input image and noiseless target image*: This is the case that was considered in the previous section.

Case II) *Noisy input image and noisy target image*: This case is more relevant in practice, as the images are generally corrupted by noise to a certain degree.

Case III) *Noisy input image and noiseless target image*: This case is considered to assess the performance of the algorithm for its compression and noise reduction capability.

PSNR values are computed with respect to the original noise free images. Fig 9 (a) shows Lena image corrupted by Gaussian white noise with PSNR 23.167 db. Fig 9 (d) shows image reconstructed by our method, with PSNR 26.7db at 0.1382 bpp when the network was trained using case I. Fig 9 (a-b) plots PSNR versus noise density for the three cases discussed above to train the network in the algorithm on image Lena. The following are the observations:

1) The algorithm using the network trained using noiseless input and noiseless target images (case I) filters out the additive noise of the input image to some extent.
2) The algorithm using the network trained using noisy input and noisy target images (case II) does not demonstrate any ability in filtering the additive noise of the input image. PSNR values are comparatively lower.
3) In the case III, when the network is trained using noisy input and noiseless target images. Obviously, the algorithm can remove the noise in the input image to some degree.

Fuzzy vector quantization algorithms perform marginally better than vector quantization algorithm in case of noisy images also. Among FVQ1, FVQ2 and FVQ3, on an average the performance of the three algorithms is similar. Fig 7 (a-b) compares various vector quantization algorithms in our method in case of noisy images.

3.4 Comparison with JPEG

In this subsection, the comparison of the performance of the proposed algorithm with the most commonly used still image encoders JPEG has been discussed. JPEG was implemented in Matlab 7. Comparison is made in terms of the visual quality and PSNR of the reconstructed images. Our encoder has achieved better performance in low bit rate environment. It is seen that our encoder consistently outperforms the JPEG encoder for all the test images. Results are tabulated in table IV. The subjective image quality of our method is better with lesser bits than JPEG and its artifacts are harder to find in the decoded image fig 8 (a) & (c). Further our method is capable of handling noise much better. Fig 10 (a-b) shows the plot of PSNR verses

increased noise strength with 0.1382 bpp in our method and in case of JPEG the bit rate varies from 0.296 to 0.4215 bpp. PSNR values are computed with respect to the original noise free images. It can be seen that our method outperforms JPEG encoder at least by 5 db. The image quality of our method is much superior from that of JPEG. The results are shown in fig 9 (b) & (d).

3.5 Comparison with JPEG2000

In this section, our algorithm is compared with still image compressor JPEG2000. JPEG2000 image encoder we used in this performance evaluation is the Jasper JPEG2000 encoder [20]. Comparison is done in terms of visual quality and PSNR of the reconstructed images versus their compression ratios. Results are in table IV. Fig 8 (a-b) shows that at similar rate for JPEG2000, the coding artifacts are slightly more visible in our method than JPEG2000. However, in case of training image "lena" the performance of our encoder matches with that of JPEG2000 encoder. Fig 11 (c-d) plots PSNR versus noise strength comparing our method and JPEG2000. Our method is capable of handling noise much better. This is due to the fact that neural network and vector quantizer is capable of filtering noise from the image corrupted with noise. The visual quality of the reconstructed image is much better in our method as compared to JPEG2000 when it is tested on noisy image. Results are shown in fig 9 (c-d).

7. Conclusion

This paper presents a neuro-wavelet based approach using various fuzzy vector quantization algorithms for codebook design for image compression at low bit rates. Various experiments were conducted on 512 x 512 x 8 bit images. Fuzzy vector quantization makes use of uncertainty to the benefit of the clustering process. Use of FVQ algorithms on the hidden layer coefficients improves the SNR and the visual quality of the reconstructed image, but it is computationally more demanding than the k-means algorithm. The average performance of FVQ1 algorithm is comparable as that of the performance of FVQ2 and FVQ3. The proposed image encoder outperforms the JPEG encoder by 40 -50% in terms of compression ratio for same PSNR for all test images. In terms of subjective quality, the proposed method is better with fewer bits as compared to JPEG but coding artifacts of our method are slightly more visible than JPEG2000 at the same bit rate. The experimental results show that the proposed technique outperforms the well known JPEG and JPEG2000 schemes in case images corrupted with noise. Our method is capable of filtering out noise from the images.

8. References

[1] V. Singh, N. Rajpal and K. S. Murthy, "A Neuro-Wavelet model using Fuzzy vector Quantization for efficient Image Compression", International Journal of Image and Graphics (IJIG), vol 9, no 2, 2009, pp 299-320.

[2] C.L.Chang and B. Grod, Direction adaptive discrete wavelet transform for image compression, IEEE Trans. Image Processing, vol. 5, may (2007), 1289–1302.

[3] K. Sasazaki, H. Ogasawara, S. Saga, J. Maeda, Y. Suzuki, Fuzzy vector Quantization of Images based on local fractal dimension, IEEE international conference on Fuzzy Systems Canada, (2006) 16-21.

[4] H Liu, l Zhai, Y Gao, W Li, J Zhou, Image compression based on biorthogonal wavelet transform, proceedings of ISCIT (2005) 578-581.

[5] L. Ma and K. Kharasani, Application of adaptive constructive neural networks to image compression, IEEE Trans. Neural networks, vol. 13, Sept (2002) 1112–1126.

[6] S. Kadono, O. Tahara, N. Okamoto, "Encoding of color still pictures wavelet transform and vector quantization", Canadian Conference on Electrical and Computer Engineering Vol 2 (2001) 931-936.

[7] D.S.Taubman and M.W. Marcellin, JPEG2000: Image Compression Fundamentals, Standards and Practice, Norvell, MA: kluwer, (2001).

[8] J.C. Burges, Y.P. Simard, and H.S. Malvar, Improving wavelet compression with neural networks, Proc. Data Compression Conf., Snowbird, Utah, Mar (2001) 486-490.

[9] K. Paliwal, V. Ramasubramian, Comments on modified K means algorithm for vector quantizer design, IEEE trans Image processing Vol 9 No 11 (2000) 1964-1967.

[10] R.D. Dony, S. Haykin, Neural network approaches to image compression, proc IEEE vol 83 Feb (1999) 288-303.

[11] J. Jiang, Image compression with neural networks–A survey, Signal processing: Image communication vol 14 no 9 (1999) 737-760.

[12] J.-S. R. Jang, C.-T. Sun, E.Mizutani, Neuro-Fuzzy and Soft Computing, Prentice Hall (1997).

[13] Karyiannis, Pai, Fuzzy vector quantization algorithms and their application in image compression, IEEE Trans. Image Processing, vol. 4, (1995) 1194–1201.

[14] R.Setiono and G.Lu, Image Compression Using a Feedforward Neural Network, Proc IEEE int conf neural network , (1994) 4761-4765.

[15] T. Denk, K.K. Parhi, and V. Cherkassky, Combining neural networks and the wavelet transform for image compression, Proc. ICASSP IEEE Int. Conf. Acoust,, Speech, Signal Processing, Minneapolis, Minn., Apr (1993) I-637–I-640.

[16] G.K. Wallace, The JPEG Still Picture Compression standard IEEE trans Consumer Electronics vol 38 no 1 Jan (1992) 18-19.

[17] S. G. Mallat, A theory for multi resolution signal decomposition: The wavelet representation, IEEE Trans. Pattern Anal. Machine Intell., vol. 11 (1989) 674–693.

[18] N. M. Nasrabadi and R. King, Image coding using vector quantization: A review, IEEE Trans. Commun., vol. 36, (1988) 957–971.

[19] J . C. Bezdek, R. Ehrlich, and W. Full, FCM: The Fuzzy c-Means Clustering Algorithm, Computers and Geosciences, Vol. 10, No. 2-3, (1984) 191-203.

[20] Jasper JPEG2000 encoder [online] available http://www.ece.uvic.ca/mdadams/jasper/

[21] Al-Shaykh and R. M. Mersereau, Lossy compression of noisy images. IEEE Trans. on Image Proc 7(12). (1998) 1641-1652.

[22] P. C. Cosman, R. M. Gray, and R. A. Olshen, Evaluation quality of compressed medical images: SNR, subjective rating, and diagnostic accuracy, *Proc. IEEE*, vol. 82, June (1994) 919–932.

[23] S. C. B. Lo, B. Krasner, and S. K. Mun, Noise impact on error-free image compression, *IEEE Trans. Med. Imag.*, vol. 9, June (1990) 202–206.

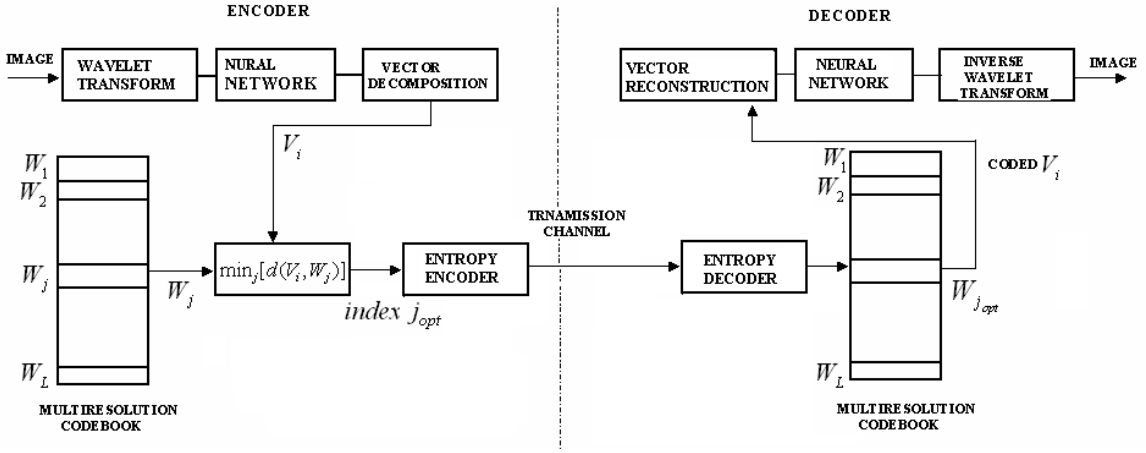


Fig 1 Encoding/decoding scheme

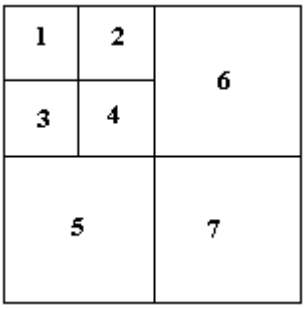


Fig 2 a Two Channel Filter Bank

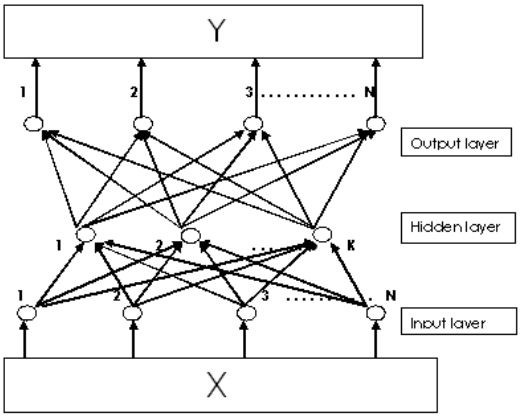


Fig 3 A multi layered neural network

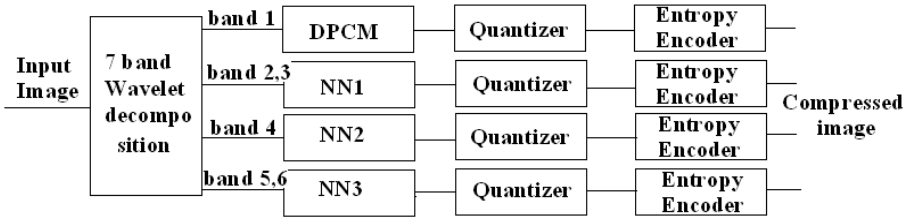


Fig 4 Complete Image compression

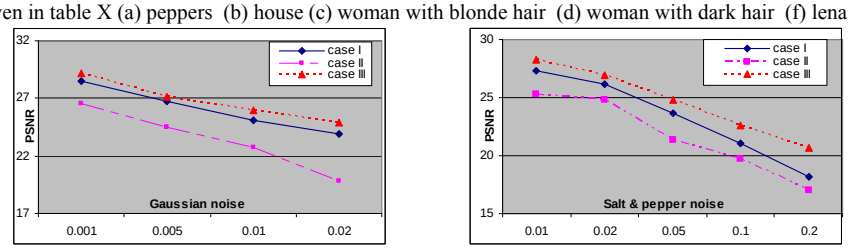
Table III PSNR(db) and compression(bpp) of different test images compressed by db18 and codebook designed using k-means, FVQ1, FVQ2 and FVQ3 algorithms

Image	PSNR With k-mean	Comp with k-mean	PSNR With FVQ1 with μ =2	Comp with FVQ1 with μ =2	PSNR With FVQ2 with λ =5	Comp with FVQ2 with λ =5	PSNR With FVQ3 with λ =5	Comp with FVQ3 with λ =5
Lena	29.402	0.1379	30.571	0.1382	30.601	0.1382	30.659	0.1384
House	31.279	0.1373	32.427	0.1387	32.525	0.1385	32.588	0.1386
Woman dark hair	34.906	0.1392	35..912	0.1395	35.992	0.1398	35.996	0.1398
Woman blonde	26.969	0.1386	28.108	0.1397	28.208	0.1398	28.215	0.1398
Peppers	28.369	0.1384	29.19	0.1393	29.32	0.1395	29.35	0.1395

(a) (b) (c)

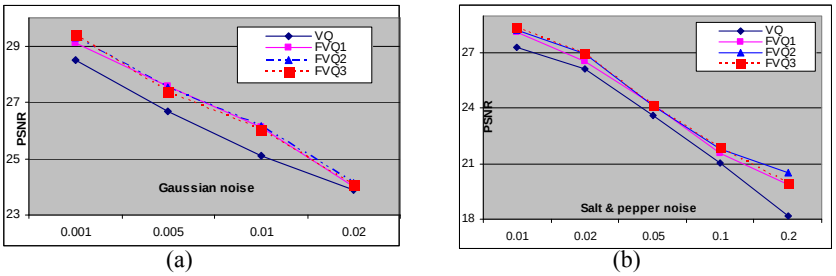
(d) (e)

Fig 5 Results of different images compressed by db18 and codebook design using FVQ3 algorithm using λ=5. compression ratio is 0.1382 bpp and PSNR values are given in table X (a) peppers (b) house (c) woman with blonde hair (d) woman with dark hair (f) lena



(a) (b)

Fig 6 PSNR versus noise strength plot of image 'lena' compressed by our algorithm with training of the neural network by three different cases discussed in section 3.3. (a) with Gaussian noise (b) with salt & pepper noise



(a) (b)

Fig 7 PSNR versus noise strength plot of image 'lena' compressed by our algorithm employing different quantization techniques (a) with Gaussian noise (b) with salt & pepper noise

Table IV Comparison of PSNR(db) and compression (bpp) of different test images compressed by our method, JPEG and JPEG2000

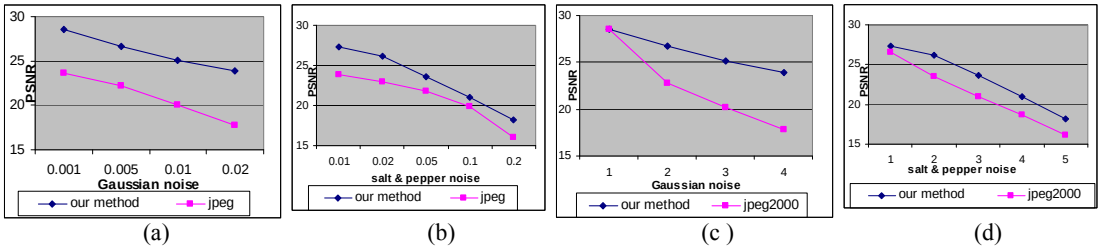
Image	PSNR With FVQ3 with λ =5	Comp with FVQ3 with λ =5	PSNR with JPEG	Comp with JPEG	PSNR with JPEG2000	Comp with JPEG2000
Lena	30.659	0.1384	30.654	0.62158	30.512	0.138
House	32.588	0.1386	32.527	0.3374	35.138	0.138
Woman dark hair	35.996	0.1398	35.421	0.217	40.395	0.138
Woman blonde	28.215	0.1398	28.209	0.2936	32.324	0.138
Peppers	29.35	0.1395	29.732	0.513	31.239	0.138

(a) (b) (c)

Fig 8 lena compressed by (a) our method (b) JPEG2000 (c) JPEG. Compression ratio and PSNR are given in table IV

(a) (b) (c) (d)

Fig 9 (a) noisy version of lena (the noise is additive white Gaussian with mean 0 and variance 0.005) Results of lena compressed by (b) JPEG with PSNR=22.143db and 0.296 bpp (c) JPEG2000 with PSNR 22.78db and 0.138bpp (d) our method with PSNR 26.7db and 0.1382bpp



(a) (b) (c) (d)

Fig 10 (a) –(b)PSNR versus noise strength comparing our method (0.1382 bpp) and JPEG (0.296 to 0.4215 bpp)
(c) –(d)PSNR versus noise strength comparing our method (0.1382 bpp) and JPEG2000 (0.138 bpp)