Journal of Intelligent Learning Systems and Applications, **2011, 3, 220-229**
doi:10.4236/jilsa.2011.34025 Published Online November 2011 (http://www.SciRP.org/journal/jilsa)

New Approaches for Image Compression Using Neural Network

Vilas H. Gaidhane[1], Vijander Singh[1], Yogesh V. Hote[2], Mahendra Kumar[3]

[1]Instrumentation and Control Engineering Division, Netaji Subhas Institute of Technology, University of Delhi, New Delhi, India; [2]Indian Institute of Technology, Roorkee, India; [3]Mahamaya Technical University, Noida, India.
Email: vilasgla98@gmail.com

Received November 15th, 2010; revised September 19th, 2011; accepted October 8th, 2011.

ABSTRACT

An image consists of large data and requires more space in the memory. The large data results in more transmission time from transmitter to receiver. The time consumption can be reduced by using data compression techniques. In this technique, it is possible to eliminate the redundant data contained in an image. The compressed image requires less memory space and less time to transmit in the form of information from transmitter to receiver. Artificial neural network with feed forward back propagation technique can be used for image compression. In this paper, the Bipolar Coding Technique is proposed and implemented for image compression and obtained the better results as compared to Principal Component Analysis (PCA) technique. However, the LM algorithm is also proposed and implemented which can acts as a powerful technique for image compression. It is observed that the Bipolar Coding and LM algorithm suits the best for image compression and processing applications.

Keywords**:** *Image Compression, Feed Forward Back Propagation Neural Network, Principal Component Analysis (PCA), Levenberg-Marquardt (LM) Algorithm, PSNR, MSE*

1. Introduction

Image compression plays an important role in communication and medical applications. The main aim of image compression is to remove the redundancy from image data in such a way that it allows the same image reconstruction at the receiver end. There are mainly two types of image compression techniques, lossy and lossless. In medical applications like X-ray and EGS images, the images are compressed by lossless compression methods because each bit of information is important. On other hand, digital or video images are compressed by lossy compression techniques [1-3]. For such type of compression, transform coding techniques like cosine transform, wavelet transform are very effective techniques, which give better results but it process the data in serial manner and hence requires more time for processing [4]. The artificial neural network is a recent tool in image compression as it processes the data in parallel and hence requires less time and therefore, it is superior over any other technique. Thus, the bottleneck type artificial neural network generally used to solve an image compression problem discussed in [5-8]. It is important to transform an image data efficiently, which is to be compressed by neural network. The data can be transformed by techniques like Principal Component Analysis (PCA), which is based on factorization techniques developed in linear algebra. Authors proposed the Bipolar Coding and Levenberg-Marquardt (LM) algorithm with back propagation neural network for image compression.

Rest of the paper is organized as follows. Section 2 provides the information regarding the architecture of bottleneck type artificial feed forward neural network. Section 3 provides the information about PCA, concept for image compression and the PCA algorithm. Section 4 presents the training algorithm for feed forward back propagation neural network. The proposed Bipolar Coding algorithm is explained in Section 5. In Section 6, the Levenberg-Marquardt algorithm is explained. Performance parameters, results and discussion are presented in Section 7. Finally, section 8 presents the conclusion and future work.

2. Artificial Neural Network Architecture

A neural network architecture as shown in **Figure 1** is used for solving the image compression problem. In this type of architecture, input from the large number of input neurons is fed to a less number of neurons in hidden layer, which is further fed to large number of neurons in

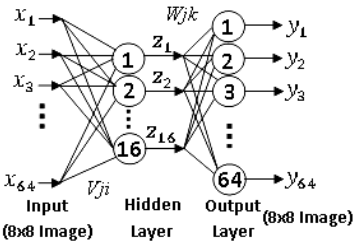

Figure 1. Bottleneck type feed forward neural network.

output layer of the network. Such type of network is referred as bottleneck feed forward neural network. One of the most important types of feed forward network is the multilayer back propagation neural network [9,10].

The neural network architecture for image compression is having 64 input neurons, 16 hidden neurons and 64 output neurons according to the requirements. The input layer encodes the inputs of neurons and transmits the encoded information to the hidden layer neurons. The output layer receives the hidden layer information and decodes the information at the output. The outputs of hidden layer are real valued and require large number of bits to transmit the data. The transmitter encodes and then transmits the output of the hidden layer 16 values as compared to the 64 values of the original image. The receiver receives and decodes the 16 hidden neurons output and generates the 64 outputs at the output layer. The techniques like Principal Component Analysis (PCA) and proposed Bipolar Coding with Linear transformations are required to incorporate with neural network to manage the input and output image data. PCA technique transforms an image data into small valued data which can be easily handled by neural network.

3. Principal Component Analysis Technique

The images obtained from the satellite are noisy and require large memory space. To compress such type of images, PCA based neural network model is used. However, adaptive neural network with PCA can also be used but it compresses the image in very less percentage [11,12]. Principal Component Analysis technique is called as Karhaunen Loeve transform, Hotelling transform or proper orthogonal transformation. It is a factorization technique which is generally used in mathematical applications. PCA technique collects the data and transforms it to the new data which results in same statistical properties. The data transformation is performed in such a way that its originality remains at the end of transformation [13].

The image data can be represented by a set of *m* vectors: $A=(a_1,a_2,\cdots,a_i,\cdots,a_m)$, where, a_i represent *n* features. The vector a_i is depending on image application. For example, in image compression each vector represents a major component of each vector features like color and size. The features of an image can be combined by considering the feature of each vector. Thus, data set ***A*** represent the features column vector *k* as:

$$C_{A,k}=\begin{bmatrix} a_{1,k} \\ a_{2,k} \\ a_{3,k} \\ \vdots \\ a_{n,k} \end{bmatrix} \tag{1}$$

This approach requires for the computation of the input data convergence matrix C_A and extraction of the eigenvalues and eigenvectors. The feature column vectors can be grouped in a matrix form for easy processing. That is,

$$C_A=[C_{a1},C_{a2},\cdots,C_{an}] \tag{2}$$

where the values of *a* and *k* are varying from 1 to *n*.

For compression of data, the component which has less importance has to be eliminated from the matrix. Therefore, the less important data is replaced by zero value. The new vectors which give the better classification properties can be form the feature column vectors $C_{a,k}$. PCA method confirms that the important data which account maximum variation in the covariance matrix to be considered. The Linear dependence between two random variables can be defined by the covariance measures. Therefore, by computing the covariance, the relationship between two data sets can be established. If, $A_i=(a_{i,1},a_{i,2},\cdots,a_{i,n})$ then the covariance is defined by equation:

$$\sigma_{A1,n}=E\{(C_{A,1}-\mu_{A,1})(C_{A,n}-\mu_{A,n})\} \tag{3}$$

$E\{\cdot\}$ is the average value of the elements of the vector and $\mu_{A,n}$ is the column vector obtained by multiplying the scalar value $E\{C_{A,k}\}$ by unitary vector. It is important to note that the covariance measures a linear relationship between the two values of sets. PCA method gives the simple solution in many applications like linear modeling, data compression, image compression, image restoration and pattern recognition etc.

The image data can also be compressed using the concept of data transformation using PCA. The PCA algorithm for data transformation and data compression is summarized in the following steps [14]:

Step 1: Obtain the feature column vector matrix C_A from the given image data.

Step 2: Obtain the covariance matrix $\sum A$.

Step 3: Using characteristic equation $(\lambda_i I-\sum A)=0$, Obtain the eigenvalues. These eigenvalues forms the matrix $\sum y$.

Step 4: Also, calculate the eigenvectors matrix *W* by

considering the eigenvalues λ_i .

Step 5: Obtain the Transformation W^T by considering the eigenvectors as their columns.

Step 6: Obtain the features vector matrix by $C_y = C_A W^T$.

Step 7: For compression of an image, the dimensionality of the new feature vector is reduce by setting small eigenvalues λ_i to zeros.

The dimensionality of input feature vector is changed by considering the small eigenvalues as zeros. The accuracy of the obtained results depends upon the selected threshold value of an eigenvalue. If the threshold value of eigenvalue is high, then the dimension of feature eigenvector is reduced more and hence the high compression can be achieved. This method considers only the data which has the maximum variation. Thus, PCA gives the simple solution to achieve high data compression, image compression and eigenface recognition. But, at high threshold eigenvalue, the loss of image data is very large. To overcome this drawback in PCA, authors suggested a bottleneck type neural network which is used to change the dimension of feature eigenvector matrix efficiently as well as reconstruction of an original image data. The architecture of such bottleneck type feed forward back propagation neural network for compression application is shown in **Figure 1**.

4. Training Algorithm for Feed Forward Neural Network

To achieve the high compression and faithful decompression of an image, the training algorithm of feed forward back propagation neural network is summarized in the following steps [15]:

Step 1: Initially, weights are required to initialize to a small random values. Each input neuron receives the input signal x_i and transmits to hidden units.

Step 2: Hidden units of neural network sums its weighted input signals along with biases as follows:

$$z_{inj} = v_{oj} + \sum_{i=1`}^{n} x_i v_{ji} \tag{4}$$

On application of activation function, the output $z_j = f\left(z_{inj}\right)$ and it sends this signal to all neurons of output layer.

Step 3: Now, output neurons ($y_k, k = 1,\cdots,m$), add all weighted inputs, $y_{ink} = w_{ok} + \sum_{j=1}^{p} z_j w_{jk}$ and applies its activation function to calculate the output

$$y_k = f\left(y_{ink}\right)$$

Step 4: Output unit compares the calculated output corresponding to a target and error is calculated as,

$$\sigma_k = \left(t_k - y_k\right) f\left(y_{ink}\right) \tag{5}$$

Step 5: The error from the output is distributed to the hidden layer, then the each hidden unit $\left(z_j, j = 1,...,n\right)$, sums its inputs from units in the layer

$$\sigma_{inj} = \sum_{k=1}^{m} \delta_j w_{jk} \tag{6}$$

The error is calculated as,

$$\sigma_j = \sigma_{inj} f\left(z_{inj}\right) \tag{7}$$

Step 6: Each output neuron (y_k), updates its bias and weights $\left(j = 1,\cdots, p\right)$ and the weights get corrected by $\Delta w_{jk} = \alpha\ \sigma_k z_j$, similarly the bias corrected by the term $\Delta w_{ok} = \alpha\ \sigma_k$.

Step 7*:* The new updated weights are represented by the following terms as;

$$w_{jk}\left(\text{updated}\right) = w_{jk}\left(\text{old}\right) + \Delta w_{jk} \tag{8}$$

$$w_{ok}\left(\text{updated}\right) = w_{ok}\left(\text{old}\right) + \Delta w_{ok} \tag{9}$$

Step 8: The weight correction term for hidden neurons,

$$\Delta v_{ij} = \alpha\ \sigma_j x_i \tag{10}$$

The bias correction term for hidden neurons,

$$\Delta v_{oj} = \alpha\ \sigma_j \tag{11}$$

Step 9: New updated weights are,

$$v_{ij}\left(\text{updated}\right) = v_{ij}\left(\text{old}\right) + \Delta v_{ij} \tag{12}$$

$$v_{oj}\left(\text{updated}\right) = v_{oj}\left(\text{old}\right) + \Delta v_{oj} \tag{13}$$

Along with PCA algorithm, the above feed forward back propagation algorithm is used to train the neural network. The accuracy of the decompression is depends upon the threshold eigenvalue at which the iteration process of learning is completed. In PCA technique, it is very difficult to obtain the correct covariance matrix. Similarly, the precise value of eigenvalues and eigenvectors is not possible due to computational error and hence the results are not satisfactory. Authors, used the neural network to minimize these errors in PCA, but the decompressed results are still not satisfactory. To overcome these problems in PCA, authors proposed another method, Bipolar Coding with linear transformation along with feed forward back propagation neural network.

5. Bipolar Coding Technique

The Bipolar Coding technique is a new approach in the field of image processing. The Bipolar Coding transformation is based on the bipolar activation function. The authors have proposed and applied this technique along with neural network for image compression. The sets of data obtained from an image are in analog form and required to convert into digital form. This conversion is possible using bipolar coding with linear scaling. The scaling has the advantage of mapping the desired range of

maximum and minimum analog value of the data set. The conversions are based on certain ranges where analog form is scaled between value 0 and 1. Thus, for converting analog values into digital form, different binary values can be assigned. In this technique, each value is converted into the range between 0 and 1 using the formula as follows [15]:

$$\delta = X_{\max} - X_{\min}$$
$$Y = \text{Intercept } C = \left(X - X_{\min}\right)/\delta \quad (14)$$

These converted values are used as the input to the neural network. Even though the neural network architecture takes input from 64 nodes down to 16 nodes, no actual compression has been occurred because unlike the 64 original inputs which are 8-bit pixel values, the outputs of the hidden layer are also real-valued, which requires possibly an infinite number of bits for transmission. True image compression occurs when the hidden layer outputs are digitized before transmission. The **Figure 2** shows a typical digitization scheme using 3 bits as reported in [16]. In this scheme, there are 8 possible binary codes: 000, 001, 010, 011, 100, 101, 110, and 111.

Each of these codes represents a range of values for a hidden unit output. For example, consider the first hidden output, when the value is between –0.1 and –0.75, then the code 000 is transmitted and when the value is between 0.5 and 0.75, the code 110 is transmitted. The concept of residual blocks conversion also used by G. Qiu *et al.* [17] but the PSNR observed was very less. The proposed Bipolar Coding technique using feed forward back propagation neural network is summarized in the following steps:

Step 1: Divide the image into small 8 × 8 chucks. These chucks are in square matrix form. It is easy to perform operation on such symmetric matrix.

Step 2: Obtain the values of pixels (0 to 255) of matrix and convert these values in the bipolar range –1 to 1. This is called as pixel to real number mapping.

Step 3: Now apply these values of bipolar range to the feed forward back propagation neural network. This neural network must have 64 input neurons as the chucks are of the size 8 × 8. Train the neural network using training algorithm as explained in Section 4.

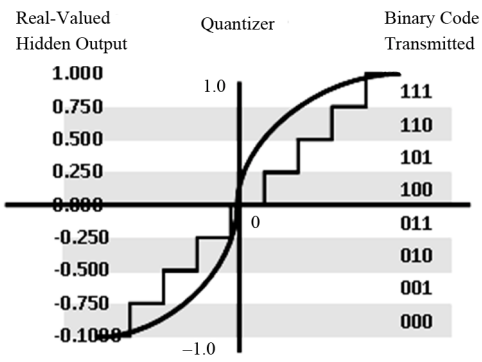

Figure 2. Digitization of hidden unit output.

Step 4: The bipolar values with weight and biases feed to the hidden layer which may have 2, 4, 8, 16, 32, and 64 hidden neurons. Convert these values in digital bits as explain in Section 5.

Step 5*:* Select the 8 × 8 chucks in a sequence after the completion of training of the neural network.

Step 6: Digital bits are now converted to real values.

Step 7: Matrix ranges bipolar values from –1 to 1 reconverted from real value to (0 to 255) pixel mapping.

Step 8: Finally, recall phase demonstrate the decompressed image (output image) at the output of the output layer of neural network. Pixel to real value and real value to pixel conversion is done during the compression and decompression process.

The proposed Bipolar Coding technique using feed forward back propagation neural network converts decimal values into its equivalent binary code and reconvert in decompression phase. The compression and quality of image depends on number of neurons in the hidden layer. The quality of output image improves and the data loss reduces as the number of hidden neurons increases.

In PCA technique, the eigenvector matrix obtained from the covariance matrix is consisting of large decimal values. The statistical calculation using these values results in erroneous results. The proposed Bipolar Coding uses the normalized –1 to 1 decimal value and its equivalent binary bits to represent the image data. The statistical calculation in Bipolar Coding becomes less complex and therefore the loss of data is reduced. Thus, this technique gives the better result as compared to PCA technique. On other hand, as it is an iterative, it requires large time for data conversion and iterations. It needs large memory space to store the subsequent results. Sometimes, if the image is of more pixel size, then system is unable to perform required number of iterations as it required the more storage space.

The output values of the neural network are used for calculation of the updated weights and biases in the next iteration. This problem of neural network learning can be considered as a function of optimization, where network is trying to determine the parameters like weights and biases to minimize the network errors. Authors proposed the use of *Levenberg-Marquardt* algorithm which acts as optimization technique for fast reduction of an error. Authors tried to use these advantages of *Levenberg-Marquardt* in image compression application. The LM algorithm overcomes the drawbacks of PCA with neural network and proposed Bipolar Coding technique.

6. Levenberg-Marquardt Algorithm

Neural networks are suitable for linear as well as highly

nonlinear functions with the adjustable biases and weights. The Levenberg-Marquardt (LM) algorithm can be used for nonlinear functions. It is very simple, but robust method for approximating the nonlinear function that locates the minimum of a multivariate function which is expressed as the sum of squares of non-linear real valued function [18,19]. LM is a combination of two methods, steepest descent and the Gauss-Newton method. The steepest decent is a first order optimization method to find the local minimum of a function. The algorithm behaves like a steepest descent method when the current solution is far from the correct one. The steepest descent method is widely popular among researchers due to its easy concept and relatively less work steps. Consider the function $F(x)$ which can be defined and differentiable within a given boundary. The negative gradient of $F(x)$ represents the directions of fastest decrease of the function. To obtain the local minimum of $F(x)$, the Steepest Descent is employed, where it uses a zigzag path from an arbitrary point X_0 and gradually slide down the gradient, until it reaches to the actual point of minimum. Mathematically, this step can be defined by iterative form as,

$$w_{n+1} = w_n - \lambda_n \nabla F\left(w_n\right) = w_n - \lambda_n g\left(w_n\right) \tag{15}$$

Here, the term $g\left(w_n\right)$ is the gradient at a given point and λ_n should be used as a step in the gradient direction. The Gauss-Newton algorithm is a method that solves the nonlinear least square problems. If m functions are given, $r_1, r_2, r_3, \cdots, r_m$ of n variables $x = x_1, x_2, x_3, \cdots, x_n$, with $m \geq n$, the Gauss–Newton method finds the minimum of the sum of squares,

$$\mathrm{Sum}\left(x\right) = \sum_{i=1}^{m} r_i^2\left(x\right) \tag{16}$$

If, $x^{(0)}$ is the initial Gauss for the minimum, then the method executes by iterations, $x^{(\mathrm{sum}+1)} = x^{\mathrm{sum}} + \Delta$. In this, Δ is the solution of the normal equations, $\left(J^{\mathrm{T}}J\right)\Delta = J^{\mathrm{T}}r$. Where, r is the vector function, J is the Jacobian matrix of r. It is slow but guaranteed correct solution. The LM algorithm incorporates the advantages of steepest descent and Gauss-Newton algorithm. Authors proposed and implemented successfully the LM mathematical technique in the application of image processing. The normal LM equation for image compression application can be modified as,

$$\left(J^{\mathrm{T}}J + \lambda I\right)\delta = J^{\mathrm{T}}E \tag{17}$$

where J is the Jacobian matrix, λ is the Levenberg's damping factor, δ is the weight update vector and E is the error vector. The $J^{\mathrm{T}}J$ matrix is called as Hessian matrix. The damping factor λ is required to adjust after each iteration and it guides the optimization process. The Jacobian is a matrix contains first derivatives of the network errors with respect to the weights and biases. It can be obtained by taking the partial derivatives of output with respect to weight as given below:

$$J = \begin{bmatrix} \frac{\partial \mathrm{F}\left(x_1,w\right)}{\partial w_1} & \frac{\partial \mathrm{F}\left(x_1,w\right)}{\partial w_2} & \cdots & \frac{\partial \mathrm{F}\left(x_1,w\right)}{\partial w_{\mathrm{n}}} \\ \frac{\partial \mathrm{F}\left(x_2,w\right)}{\partial w_1} & \frac{\partial \mathrm{F}\left(x_2,w\right)}{\partial w_2} & \cdots & \frac{\partial \mathrm{F}\left(x_2,w\right)}{\partial w_{\mathrm{n}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathrm{F}\left(x_{\mathrm{N}},w\right)}{\partial w_1} & \frac{\partial \mathrm{F}\left(x_{\mathrm{N}},w\right)}{\partial w_2} & \cdots & \frac{\partial \mathrm{F}\left(x_{\mathrm{N}},w\right)}{\partial w_{\mathrm{n}}} \end{bmatrix} \tag{18}$$

where $\mathrm{F}\left(x_i,w\right)$ is the network function evaluated using the weight vector w for the i^{th} input vector. Hessian doesn't need to be calculated as it is approximated by using the Jacobian matrix. This approximation of the Hessian holds good if the residual errors at the solution are small enough. The general Levenberg-Marquardt algorithm consists of the following steps:

Step 1: Obtain the Jacobian matrix J. It is recommended to use finite difference or chain rule for calculation of J.

Step 2: Obtain the error gradient $g = J^{\mathrm{T}}J$

Step 3: Now, Hessian matrix can be approximated using the cross product of Jacobian as: $H = J^{\mathrm{T}}J$

Step 4: The value of δ can be obtained by solving the equation $\left(J^{\mathrm{T}}J + \lambda I\right)\delta = J^{\mathrm{T}}E$.

Step 5: Use the value of δ to update the network weights w.

Step 6: Using the updated weights w, recalculate the sum of squared errors.

Step 7: If sum of the squared errors has not further reduced then discard the new weights values and increase the value of λ.

Step 8: Else decrease λ and calculate new values of w using δ till the error reaches at the desired value and stop.

Here, λ start from very small value such as 0.1 and if it becomes large then iterations stops and the value of λ again decreases.

In this algorithm, if the damping used at one iteration reduces the error, the damping is divided by a reduction factor before the next iteration and the convergence speed up towards the best result. If the error increases, then the damping multiplied by the factor, ensure the correct result. Thus, the algorithm switches from one to another smoothly. If the proper value of damping selected in LM, then its iterations completed within the short duration and produced the best results at decompressed phase. The LM algorithm thus required the less number of iteration and less time as compared to PCA technique as well as the Bipolar Coding technique. The memory requirement in LM method is also less as the

number of iteration reduces.

7. Experimental Evaluation and Results

The quality of an image is measured using the parameters like Mean Square Error (MSE) and Peak Signal to Noise ratio (PSNR). MSE and PSNR are the parameters which define the quality of an image reconstructed at the output layer of neural network. The MSE between the target image and reconstructed image should be as small as possible so that the quality of reconstructed image should be near to the target image. Ideally, the mean square error should be zero for ideal decompression. The compression-decompression error is evaluated by comparing the input image and decompressed image using normalized mean square error formula [16].

$$E = \left(t - y_{\mathrm{k}}\right)^2 \tag{19}$$

where t is the target value and y_{k} is the output of the neural network.

An alternative to MSE, as indication of decompressed image quality, peak signal to noise ratio (PSNR) is also introduced which is defined as [17]:

$$\mathrm{PSNR}\ (\mathrm{dB}) = 10 \cdot \log_{10}\left(\frac{L^2}{\mathrm{MSE}}\right)(\mathrm{dB}) \tag{20}$$

where L is the maximum value of pixels of an image. The peak signal to noise ratio depends on the maximum value of pixels of an image and MSE. Therefore, the PSNR also defines the quality of image. The PSNR of reconstructed image should be as high as possible and MSE as minimum as possible so that the almost same quality of image can be obtained after decompression. For various experimentation in different techniques described above, *Leena* and *Lussy* images are considered which are shown in **Figure 3(a)**. Specifications of images are obtained by reading images in Adobe Photoshop 7.0.1 image editing software.

- Specifications of image: *Leena*

Dimensions = 256 × 256 pixels, Size: 64 K,
Resolution = 28.346 pixels/cm.
Width = 7.57 cm, Height = 6.96 cm.

- Specifications of image: *Lussy*

Dimensions = 32 × 32 pixels, Size: 1 K,
Resolution = 28.346 pixels/cm.
Width = 1.13 cm, Height = 1.13 cm

The input images *Leena* and *Lussy* are first converted into 8 × 8 pixels blocks. The image compression is experimented with various numbers of neurons in the hidden layer along with the different parameters of the neural network. The neural network configuration with one hidden layer and different number of neurons in hidden layer as 2, 4, 8, 16 & 32 can be easily represented by binary bits respectively. Therefore 2, 4, 8, 16 & 32 hidden neurons are considered for experiments while 64 input neurons and 64 output neurons are used according to the requirement of selected input image pixels blocks size.

A number of experiments are carried out to test the PCA algorithm using feed forward back propagation neural network. For various experiments a common images, *Leena* and *Lussy* are considered which are shown in **Figure 3(a)**. The 1000 epochs are considered, as the best results are obtained at 1000 epochs.

The results of 256 × 256 pixels *Leena* and 32 × 32 pixels *Lussy* images using PCA technique are summarized in **Table 1**. These are the measurements for constant values of learning rate α = 0.2 and momentum factor (m.f.) = 0.5. Decompressed image for 32 hidden neuron results into good quality image compared to 2, 4, 8 and 16 hidden neurons. The decompressed images for 32 hidden neurons are shown in **Figure 3(b)**.

From the results, it is observed that the decompressed image is distorted due to the loss of data during computation. Also, the error remain constant due to saturation of the network and therefore decompressed images for 2, 4, 8, 16 and 32 hidden neurons is also approximately same which are shown in **Figure 3(b)**. **Figure 4** shows that the error saturates and becomes constant which leads to more distortion in the output.

Now, consider the *Leena* and *Lussy* as input images with same specifications for proposed Bipolar Coding with linear scaling technique. **Table 2** shows experimental results for Bipolar Coding with linear scaling algorithm. These are the measurements for constant values of learning rate α = 0.2 and momentum factor (m.f.) = 0.5. The 1000 epochs are considered for *Leena* and *Lussy* input images. The image in **Figure 3(c)** shows the decompressed images for 32 hidden neurons. Also, the variation of an error with respect to number of epoch performed is shown in **Figure 5**.

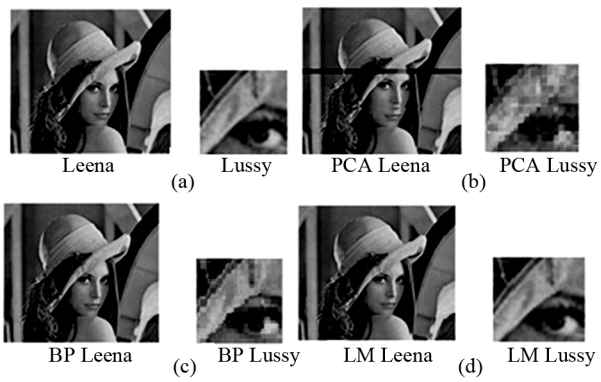

Figure 3. (a) Original Input images for experiments; (b) Decompressed images using PCA technique; (c) Decompressed images using proposed Bipolar Coding technique; (d) Decompressed images using Levenberg-Marquardt algorithm.

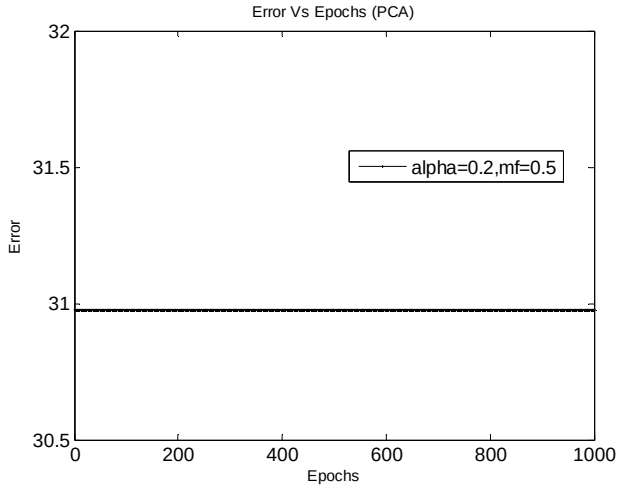

Figure 4. Error versus Epochs for PCA technique.

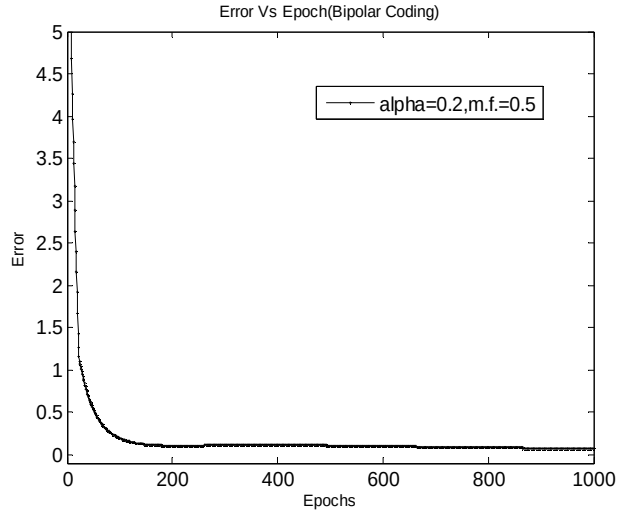

Figure 5. Error versus Epochs for proposed Bipolar Coding technique

Table 1. Results for *Leena* and *Lussy* image using PCA technique, α = 0.2 and m.f. = 0.5.

Hidden	Leena			Lussy	
Neurons	*Epoch*	*PSNR* (dB)	*Error*	*PSNR* (dB)	*Error*
2	1000	29.9798	30.9755	30.2798	32.9755
4	1000	29.3997	30.9775	30.3047	30.9775
8	1000	29.3559	30.5693	30.3554	30.9693
16	1000	30.1157	30.5375	30.3657	30.9775
32	1000	30.3463	30.5070	30.3863	30.9770

Table 2. Results for *Leena* and *Lussy* image using Bipolar Coding technique (α = 0.2 and m.f. = 0.5).

Hidden	Leena			Lussy	
Neurons	*Epoch*	*PSNR* (dB)	*Error*	*PSNR* (dB)	*Error*
2	1000	42..412	20.38	68.9702	0. 2641
4	1000	48.7998	19.645	72.0297	0.0082
8	1000	59.5776	9.6119	72.7644	0.0041
16	1000	64.6756	0.1232	85.5733	5.3E-03
32	1000	68.4563	0.0089	88.9967	1.9E-05

From these results, it is observed that the Bipolar Coding with linear scaling algorithm gives better resuslts than statistical PCA method. But in Bipolar Coding, if the size of data set is large (*Leena* image), then iterations stops without minimizing the error. This occurs due to the saturation of the network. Also, it requres the large memory area to store the iterated results. Such type of problem of neural network learning can be overcome by Levenberg-Marquardt algorithm. The results further can be improved and errors can be minimised by applying the Levenberg-Marquardt algorithm.

The experiments are carried out for images shown in **Figure 3(a)** with same specifications for 2, 4, 8, 16 and 32, hidden neurons and 1000 epochs. In LM technique, error is very less and PSNR is very high as compared to PCA and Bipolar Coding proposed technique. Also, it is observed that the decompressed images are approximately same as the target images (input images). In **Figure 3(d)**, LM Leena and LM Lussy images show the decompressed images for 32 hidden neurons for 1000 epochs. Levenberg-Marquardt algorithm results are summarized in **Table 3**. **Figure 6** shows the relation between the variation of error and epochs performed using LM algorithm. **Figure 7** shows the relative variation of the PSNR (dB) with respect to epochs performed. If *Leena* image is considered as an input image for Levenberg-Marquardt algorithm, due to large size of image, the matrix size is also large and it is not possible to perform more iteration. Therefore, it is convenient to perform 200 - 1000 iterations for large images. Even with the small number of iterations, the results using Levenberg-Marquardt algorithm for large image are quite good and the error is very less as compared to the other methods.

The momentum factor and learning rate parameter decides the speed and error of the feed forward back propagation neural network. If momentum factor and learning rate is small then training of the neural network is slow and therefore required more time for compression.

On other hand, if both parameters chosen with high value then the oscillations occurs in the network. Thus, the proper selection of these parameters is an important task in PCA, Bipolar Coding and LM technique. Typically, momentum factor value is $0 \le m.f. \le 0.9$ and learning rate value is $0.05 \le \alpha \le 0.8$ is considered. **Table 4** shows the statistical comparison of all three techniques at different values of learning rate and momentum factor.

All the discussed techniques can also be compared using the parameter like speed with respect to epochs performed. **Table 5** shows the comparison between the different techniques. These reading are for Hidden neurons = 32, α = 0.2 and m.f. = 0.5. *Lussy* image is considered for experimentations. From **Table 5**, it is observed that PCA technique requres more time as compared to Bipolar coding and LM technique. It is also seen that the LM technique is faster than PCA and Bipolar coding technique. For following results, the system with following specifications is used: Intel(R) Core™ 2 Duo CPU, T5670 @1.80 GHz, 789 MHz Processor.

Table 3. Results for *Leena* and *Lussy* image using LM Algorithm technique (α = 0.2 and m.f. = 0.5).

Hidden		Leena		Lussy	
Neurons	*Epoch*	*PSNR* (*dB*)	*Error*	*PSNR* (*dB*)	*Error*
2	1000	117.9094	1.1E-07	138.3854	9.8E-10
4	1000	119.7014	7.2E-08	140.0208	6.7E-10
8	1000	130.045	6.7E-09	140.0228	6.7E-10
16	1000	149.9405	6.9E-11	148.9903	8.5E-11
32	1000	150.017	6.7E-11	151.1854	5.1E-11

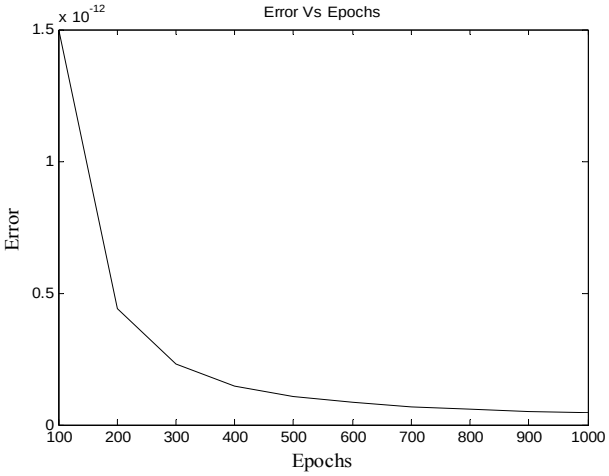

Figure 6. Error versus Epochs for LM technique.

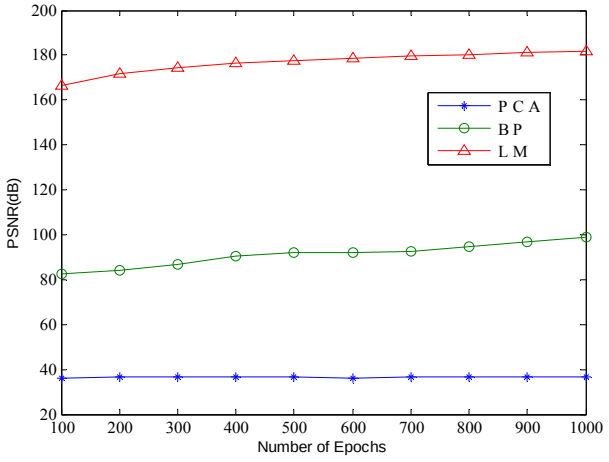

Figure 7. PSNR versus Epochs for PCA, Bipolar and LM technique.

Table 4. Comparison of PSNR (dB) and Error for PCA, Bipolar Coding and Levenberg-Marquardt techniques.

Epochs	α = 0.8, Momentum factor (m.f.) = 0.2						α = 0.5, Momentum factor (m.f.) = 0.4					
	PCA		Bipolar coding		LM Algorithm		PCA		Bipolar coding		LM Algorithm	
	PSNR	*Error*	*PSNR*	*Error*	*PSNR*	*Error*	*PSNR*	*Error*	*PSNR*	*Error*	*PSNR*	*Error*
100	36.3038	30.9795	82.5924	0.3039	166.40	1.50E–12	36.3882	32.5755	85.237	0.2241	166.34	1.58E-12
200	36.3723	30.9775	84.0024	0.2584	171.73	4.41E–13	36.3993	32.5693	88.642	0.1514	171.89	4.40E-13
300	36.4023	30.9774	87.0002	0.1707	174.57	2.30E–13	36.4013	32.9774	90.111	0.1279	174.73	2.29E-13
400	36.4085	30.9774	90.3853	0.1239	176.43	1.48E–13	36.3661	32.543	91.063	0.1146	176.59	1.49E-13
500	36.5692	30.5692	92.0072	0.1028	177.79	1.10E–13	36.392	32.9837	93.416	0.0874	177.95	1.09E-13
600	36.3392	30.9794	92.0948	0.1018	178.87	8.48E–14	36.3246	32.9775	94.667	0.0772	178.99	8.58E–14
700	36.4301	30.9794	92.6368	0.1073	179.71	7.10E–14	36.3413	32.5693	95.206	0.0711	179.88	7.00E–14
800	36.3971	30.5692	94.5684	0.0765	180.46	5.91E–14	36.3444	32.5692	96.145	0.0638	180.62	5.90E–14
900	36.3961	30.9834	96.8211	0.0592	181.11	5.09E–14	36.325	32.9774	96.967	0.0620	181.26	5.08E–14
1000	36.3734	30.9774	98.8979	0.0522	181.69	4.45E–14	36.3847	32.5692	98.606	0.0041	181.84	4.44E–14
1500	36.4723	30.7765	104.342	0.0034	184.55	2.28E–14	36.3661	32.9774	105.251	0.0024	184.79	2.26E–14
2000	36.4743	30.6795	106.544	0.0027	185.65	1.90E–14	36.4323	32.4755	107.625	0.0017	185.77	1.80E–14
2500	36.7723	30.5885	107.656	0.0019	187.11	1.30E–14	36.5423	32.5664	108.124	0.0011	187.25	1.29E–14
3000	36.7973	30.5705	109.887	0.0008	188.32	9.75E–15	36.4973	32.5554	110.621	0.0005	188.85	9.72E–15

Table 5. Comparison of execution time required for PCA, Bipolar Coding and Levenberg-Marquardt techniques.

Epoch	Time(Min:Sec:div)		
	PCA	*Bipolar Coding*	*LM*
100	0:13:04	0:10:25	0:02:53
200	0:17:07	0:12:36	0:08:11
300	0:21:15	0:14:07	0:11:08
400	0:32:51	0:18:44	0:15:26
500	0:41:20	0:23:04	0:18:04
600	0:48:36	0:37:07	0:21:56
800	1:10:15	1:05:10	0:28:15
1000	1:25:53	1:21:14	0:35:10

8. Conclusions

In this work, PCA, proposed Bipolar Coding and LM technique, based on artificial neural network are applied for image compression application. In PCA technique, the accuracy of the results obtained depends upon the threshold value of eigenvalue at which the iteration process of learning is terminated. Also, some of the information below the threshold value is removed or replaced by zero and therefore more information is removed from the feature vector matrix and hence from image data. Thus, the reconstructed image result is not satisfactory as well as the convergence speed is very slow.

The Bipolar Coding Technique and LM algorithm are proposed and implemented for image compression and got the satisfactory results as compared to PCA technique. The Bipolar coding network is trained with the small 8 × 8 blocks of image and tested. It is observed from the results that using this method, a good quality of decompressed image is obtained. It has high PSNR and very less error. Thus, this method achieves high compression. But, the neural network is trying to determine the updated weights and biases in each step to minimize the systems errors. This is step by step process that requires more time and more memory space to store the subsequent results. To overcome these problems and to improve the results, the mathematical Levenberg-Marquardt algorithm is proposed. It is observed from the experimental results that the image compression using Levenberg-Marquardt algorithm performs better than the PCA and Bipolar coding and is having more convergence speed. It is also seen that the Levenberg-Marquardt algorithm suits the best for small as well as large image compression. This algorithm is fast in operation as well as it requires less memory to store the results. The Bipolar Coding technique and LM algorithm can be applied for many image processing applications where the data is highly nonlinear. The images taken by satellite or remotely sensed images are generally noisy and therefore it is necessary to perform lossless compression. The proposed methods are most suitable for such compression as PSNR is very high and less error. The adaptive nature of the proposed methods (Bipolar Coding and LM algorithm) results into variety of applications in image processing.

9. Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive comments and suggestions. They also want to express their gratitude to Prof. J. R. P. Gupta and Prof. A. P. Mittal who provided the infrastructure and support for this work.

REFERENCES

[1] E. Watanabe and K. Mori, "Lossy Image Compression Using a Modular Structured Neural Network," *Proceedings of IEEE Signal Processing Society Workshop*, Washington DC, 2001, pp. 403-412.

[2] M. J. Weinberger, G. Seroussi and G. Sapiro, "The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS," *IEEE Transaction on Image Processing*, Vol. 9, No. 8, 2000, pp. 1309-1324.

[3] V. H. Gaidhane, Y. V. Hote and V. Singh, "A New Approach for Estimation of Eigenvalues of Images," *International Journal of Computer Applications*, Vol. 26, No. 9, 2011, pp. 1-6.

[4] S.-G. Miaou and C.-L. Lin, "A Quality-on-Demand Algorithm for Wavelet-Based Compression of Electrocardiogram Signals," *IEEE Transaction on Biomedical Engineering*, Vol. 49, No. 3, 2002, pp. 233-239.

[5] O. A. Ahmad and M. M. Fahmy, "Application of Multilayer Neural Networks to Image Compression," 1997 *IEEE International Symposium on Circuits and Systems* (ISCAS), Hong Kong, 1997, pp. 1273-1276.

[6] L. Ma and K. Khorasani, "Application of Adaptive Constructive Neural Networks to Image Compression," *IEEE Transactions on Neural Networks*, Vol. 13, No. 5, 2002, pp. 1112-1126.

[7] Q. Du, W. Zhu and J. E. Fowler, "Implementation of Low-Complexity Principal Component Analysis for Remotely Sensed Hyper-Spectral Image Compression," *Proceeding of IEEE Workshop on Signal Processing Systems*, Shanghai, October 2007, pp. 307-312.

[8] I. Vilovic, "An Experience in Image Compression Using Neural Networks," *Proceeding of* 48th *International Symposium Elmar*, Zadar, 2006, pp. 95-98.

[9] M. Daszykowski, B. Walczak and D. L. Massart, "A Journey into Low-Dimensional Spaces with Autoassociative Neural Networks," *Talanta*, Vol. 59, No. 6, 2003, pp. 1095-1105.

[10] V. Gaidhane, V. Singh and M. Kumar, "Image Compression Using PCA and Improved Technique with MLP Neural Network," *Proceedings of IEEE International Con-*

ference on Advances in Recent Technologies in Communication and Computing, Kottayam, 16-17 October 2010, pp. 106-110.

[11] R. C. Gonzalez, R. E. Woods and S. L. Eddins, "Digital Image Processing Using MATLAB," Pearson Edition, Dorling Kindersley, London, 2003.

[12] J.-X. Mi and D.-S. Huang, "Image Compression Using Principal Component Analysis Neural Network," 8th *IEEE International Conference on Control*, *Automation*, *Robotics and Vision*, Kunming, 6-9 December 2004, pp. 698-701.

[13] S.-T. Bow, B. T. Bow and S. T. Bow, "Pattern Recognition and Image Processing," Revised and Expanded, 2nd Edition, CRC Press, Boca Raton, 2002.

[14] M. Nixon and A. Aguado, "Feature Extraction & Image Processing," 2nd Edition, Academic Press, Cambridge, 2008, pp. 385-398.

[15] S. N. Sivanandam, S. Sumathi and S. N. Deepa, "Introduction to Neural Network Using MATLAB 6.0," 2nd Edition, Tata Mc-Graw Hill Publication, Boston, 2008.

[16] A. Laha, N. R. Pal and B. Chanda, "Design of Vector Quantizer for Image Compression Using Self Organizing Feature Map and Surface Fitting," *IEEE Transactions on Image Processing*, Vol. 13, No. 10, October 2004, pp. 1291-1303. doi:10.1109/TIP.2004.833107

[17] G. Qiu, T. J. Terrell and M. R. Varley, "Improved Image Compression Using Back Propagation Networks," In: P. J. G. Lisbao and M. J. Taylor, Eds., *Proceeding of the Workshop on Neural Network Applications and Tools*, IEEE Computer Society Press, Washington DC, 1994, pp. 73-81.

[18] V. Singh, I. Gupta and H. O. Gupta, "ANN Based Estimator for Distillation Using Levenberg-Marquardt Approach," *Engineering Applications of Artificial Intelligence*, Vol. 20, No. 2, 2007, pp. 249-259. doi:10.1016/j.engappai.2006.06.017

[19] M. I. A. Lourakis, "A Brief Description of the Levenberg-Marquardt Algorithm Implemented by Levmar," *Foundation of Research and Technology*, Vol. 4, 2005, pp. 1-6.

