International Journal on Emerging Technologies **11**(1): 311-318(2020)

ISSN No. (Print): 0975-8364
ISSN No. (Online): 2249-3255

Medical Image Compression using ANN Model

M. Laxmi Prasanna Rani[1], G. Sasibhushana Rao[2] and B. Prabhakara Rao[3]
[1]Assistant Professor, Department of Electronics and Communication Engineering,
MVGR College of Engineering, Vizianagaram (Andhra Pradesh), India.
[2]Professor, Department of Electronics and Communication Engineering,
AU College of Engineering (A), Andhra University, Visakhapatnam (Andhra Pradesh), India.
[3]Professor, Department of Electronics and Communication Engineering,
JNTUK, Kakinada (Andhra Pradesh), India.

(Corresponding author: M. Laxmi Prasanna Rani)
(Received 01 November 2019, Revised 30 December 2019, Accepted 02 January 2020)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: The medical images of CT, MRI and PT scan are the digital form of human body pictures. For storage and successful transmission of these images, there is a need to reduce the size of these images using dimensionality reduction or compression techniques. Image compression plays an imperative role within the field of biomedical analysis to save the storage space for the transmission of medical images to experts for better diagnosis of diseases. The compression of images is achieved using some compression techniques and obtain the reconstructed image with considerable loss of image quality. However, in medical applications, high image quality reconstructed image in the region of interest for the diagnosis of critical diseases is essential. In this paper, Feed Forward Back Propagation Artificial Neural Network (FFBPANN) model with different back propagation algorithms is used to increase the quality of the image by giving better compression ratio and less convergence time. This paper discusses the application of the FFBPANN architecture with Huffman encoding techniques to compress brain MRI images. This network provides more Peak Signal to Noise Ratio (PSNR), less Mean Squared Error (MSE) for the compression Ratio (CR) of 4:1 using Levenberg–Marquardt back propagation algorithm compared to the gradient and conjugate gradient training algorithms.

Keywords: Compression, ANN, LM, PSNR, MSE, SSIM.

I. INTRODUCTION

Nowadays, huge data files consisting of images, video files, etc., are a major hurdle in managing the space in system. Compression of these images is an essential process for creating file sizes of the images with different resolutions which are manageable and communicable. The file size to store the image is reduced by decreasing the number of bits per pixel. The techniques of image compression are most commonly used in the data storage, printing and telecommunication industry. Nowadays, digital applications like high definition television, satellite remote sensing, fax transmission, etc., use image compression techniques for better transmission and storage of data [1].

Acquiring and transmitting huge set of images is required for diagnosis of the diseases in healthcare applications, which is now a major field of expansion due to the technological advancement in medical electronics. These images have to be stored for reference and research purposes without placing large loads on system servers. To save storage space for these medical images and to send these images to multiple physicians for diagnosis, compression of these images is necessary using different types of compression techniques [2]. At the receiver, compressed images have to be reconstructed with high quality without much loss of content in image, whenever they are needed to be viewed for the diagnosis of diseases.

The lossy and lossless methods are two approaches to compress images. Lossy methods provide considerable loss of quality of images with good compression used in digital camera, networking, mobiles etc. Lossless techniques produce better image quality and used in telemedicine applications, satellite communications, etc.

For compression of these images, there are different spatial techniques based on the pixel information of images, transform techniques based on the transform of images like DCT (Discrete Cosine Transform), DWT (Discrete Wavelet Transform), various optimization techniques and training, learning algorithms using Artificial Neural Networks (ANNs).

ANNs have the ability to acquire more inputs, process them to reduce hidden and get output by learning and model non-linear, complex relationships. Hence, ANN model is used in this paper to provide better image compression with good quality of reconstructed image. ANN concept is similar to concept of functioning of human brain. This network with different weights and biases can be treated as a replica of a huge number of neurons in human brain. ANNs function like the human brain to attain required computational strength [3].

Ahamed and Chandrashekarappa (2014) implements Artificial Neural Network using Back Propagation Technique for Compression and Decompression of test image like lena image [8]. Anusha *et al.*, (2014) explains the model of Neural Image Compression using Gradient Decent Technology with weight optimization using Genetic Algorithm for the images of Lena, Peppers and

boat [13]. Charles *et al.*, (2010) developed an adaptive method for image compression based on complexity level of the image and modification on levenberg-marquardt algorithm for the test image like cameraman image [16]. This paper presents FFBPANN model with back propagation algorithms of gradient descent, conjugate gradient and quasi newton to compress and to reconstruct of images at the output. The values of PSNR, MSE, SSIM, etc., obtained using different training algorithms with different hidden nodes are compared. The encoding technique of Huffman encoding is used after hidden layer to get better compressed output with fewer bits per pixel. The encoding, simulation, and training times for different training algorithms are also observed. The organization of this paper is such that the research method includes the architecture, functioning of ANN and the methodology for compression of images with different training algorithms are explained in section II. Section III presents results and discussion of different training algorithms. The conclusions are given in section IV.

II. MATERIALS AND METHODS

Artificial Neural Networks (ANNs): ANNs involve elementary components operating in parallel. It is operated similar to the functioning like a human brain. The novel structure of processing of information is the vital part of this network. ANNs are used to resolve the specific problems using interconnected processing components [3]. The adjustments of the inter connections existing between the neurons is called learning in biological systems. Based on learning, there are many fields such as neurocomputing, parallel distributed processing, machine learning algorithms, natural intelligent systems, and artificial neural networks to solve specific problems.

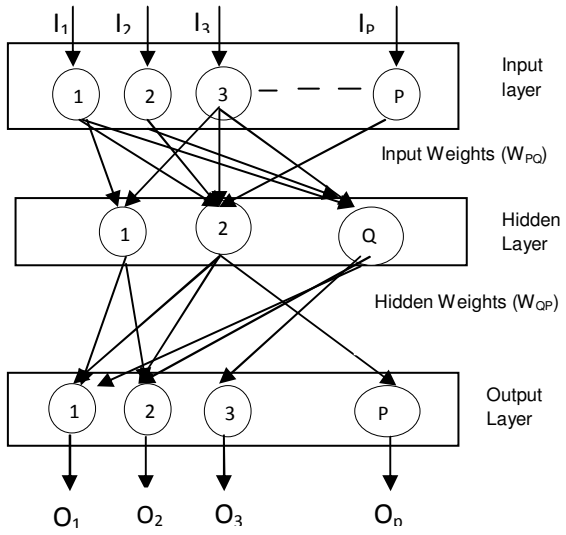

Fig. 1. Arrangement of ANN with input, hidden and output layers.

ANN architecture consists of input and output layers with hidden layer. The elements of each layer are called neurons/ nodes and these are simulated within dedicated hardware and sophisticated software [4-5]. The structure of ANN is shown in Fig. 1 below, consists of one input layer, one output layer, and one hidden layer. The inputs I_1, I_2, I_3, … I_p of P components given as neurons to the input layers. The weights W_{PQ}, W_{QP} are the weights of input and hidden layers respectively. The components O_1, O_2, O_3,…O_P are the reconstructed outputs from the output layer.

Back Propagation Artificial Neural Network (BPANN): In this BPANN structure, both the input and output layers are fully connected via hidden layer [7]. The difference between the actual output and the targeted output is referred as error. This error is reduced using back propagation by updating the weights of both input and hidden layers in backward direction. This process made the output from the output layer is closer to the targeted output. Thus this network is used to reduce the error and produce good compression with more accuracy [8]. This weight updating process and the methodology to reduce the error using BPANN architecture is explained below.

Methodology: The FFBPANNs consists of forward pass and backward pass throughout the network in forward and backward directions. The data transmitted in forward direction through the input and hidden layers and get actual response at the output layer. The fixed weights are used at both input, hidden layer during the forward direction and these are updated during the backward pass to minimize the error. The output is back propagated of the network to minimize error by updating the weights of input and hidden layers. The inputs at input layer are compressed at the hidden layer depending on the number of nodes in the hidden layer. The compression ratio is defined as the ratio of number of nodes at input layer to the number of nodes at hidden layer. This output at the hidden layer is encoded with Huffman method to reduce the number of bits. Thus, FFBPANNs are used to attain image compression and decompression. The different back propagation algorithms are applied to ANN architecture using various activation functions to get better image compression [6]. The steps involved in the image compression and reconstruction using FFBPANN are shown in Fig. 2.

Image pre-processing: The process getting the required format of input image from the original image to the ANN architecture is called preprocessing. Initially, the image is separated into blocks and each block consists of 8 × 8 pixels. Each block of 8 × 8 pixels can be converted into column vector of 64 × 1 elements. Using normalization process, each pixel value is divided by 255.

Defining a Network: After preprocessing of an image, it can be given to the input of the network structure for the compression of images. The column vector of 64 × 1 elements of each block is applied to the input nodes from node 1 to node P at input layer and this is input to neural network. The output can be obtained at output layer which is reconstructed image. Image compression is obtained by taking less number of hidden neurons/ nodes (Q) compared to input and output neurons/nodes(P).

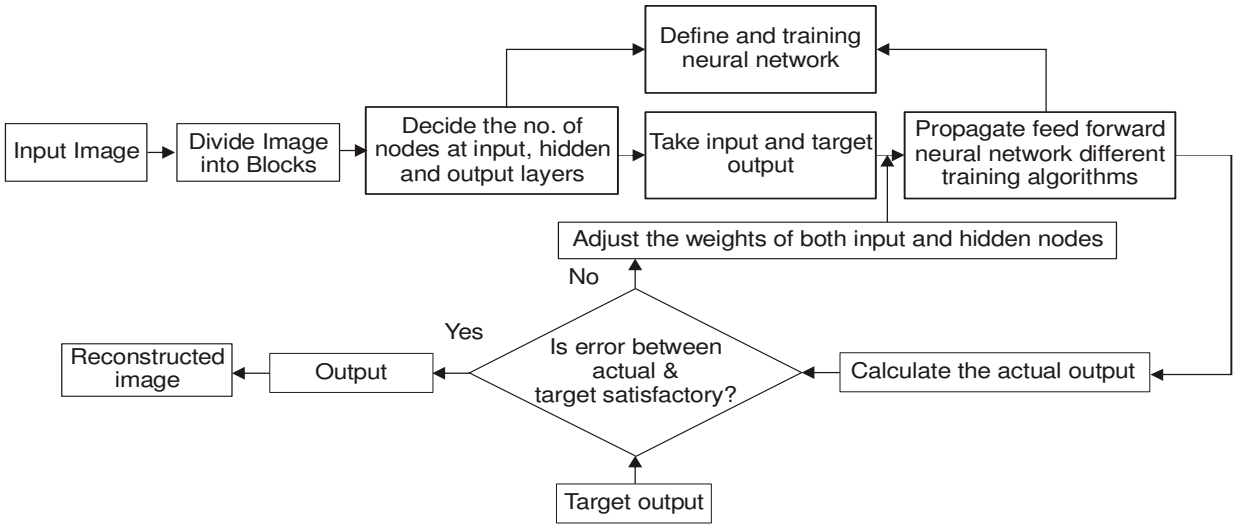

Fig. 2. The procedure involved in the process of compression and reconstruction of images using FFBPANN Model.

The output of hidden layer is the sum of products of inputs at input layer with their corresponding weights with non-linear activation function. At hidden layer, the compressed form of image is obtained. The output of hidden layer with their respective weights applied to output layer through the nonlinear activation function to get decompressed image. The process of compression of images using ANN model is similar to general image compression and this network structure is shown in Fig. 3.

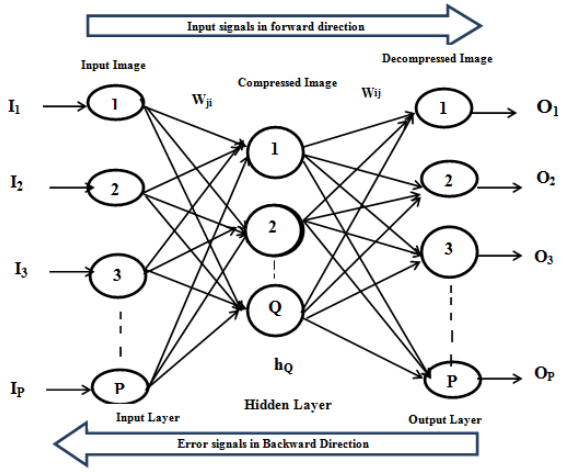

Fig. 3. FFBPANN Architecture for image compression and decompression.

The encoder at the transmitter encodes the output from the hidden layer using Huffman encoding to reduce the number of bits and then transmits the encoded output to the receiver. The receiver receives and decodes the hidden layer outputs to generate reconstructed outputs. Each Pixel consists of 8 bits, is fed into each input node and the same pixels with les number of bits will be the output after the compression has done. This structure of ANN changes as the nodes varies from P down to Q of input to hidden layer respectively to achieve compression. ANNs, defined above are categorized either as linear or nonlinear depending upon the activation function employed at the hidden and output layers. Generally, Log-sigmoid function of nonlinear is commonly used as activation function in back propagation network due to its differentiable property [8-9]. Generally, nonlinear active functions are used at hidden layer and linear functions like tangent functions at the output layer. In general, both linear and nonlinear problems can be solved by nonlinear transfer functions than linear ones. The non-linear activation/transfer function of Log-sigmoid is used at the input of hidden layer, and the activation function is s given in Eqn. (1).

$$f(X) = \frac{1}{1+\exp(-X)} \tag{1}$$

The output bj of jth neuron in the feed forward network of hidden layer is given by the Eqn. (2).

$$b_j = h\left(\sum_{1}^{Q} W_{ij} X_i + c_j\right) \tag{2}$$

where b_j is output after input layer with their strength and weights of respective nodes. h is the activation function of hidden layer. W_{ij} is the weight of input nodes. c_j is the bias of the hidden layer, and hidden nodes are Q. The reconstructed output is specified by the equation given below

$$m_k = g\left(\sum_{1}^{p} W_{kl} b_k + c_k\right) \tag{3}$$

where, m_k is the actual output at output layer, g is the activation function at output layer, W_{kl} is the weight linking the kth hidden node to the l^{th} output node. c_k is the bias of the kth neuron of the output layer and P is the number of nodes in the output layer.

Training a Network: After defining and creating ANN model, this model can be trained with training algorithms

by updating input and hidden weights. Input and hidden weights are chosen randomly.

After selecting the weights, training and learning will start. Supervised and unsupervised learning are two methods of learning used to train a network. Supervised learning provides desired outputs for the respective inputs providing the network. Unsupervised learning involve with the inputs and no corresponding output data [10-11].

Training Algorithms: After the selection of weights, different training algorithms are used to train a neural network. BP algorithm is used to make target output is nearer to actual output by updating weights of the input and hidden layers respectively. These weights are updated using the equation given below.

$$W_{t+1} = W_t + \beta.\delta W \quad (4)$$

Where β is the learning coefficient. The value of is varied to make actual output equal to target output using different training algorithms [12]. The network is trained using these algorithms and get the training time and quality metrics for a given number of training epochs. The algorithms for the compression are Gradient Descent algorithms with four training functions, Conjugate Gradient algorithms with three training functions and also Quasi-Newton algorithms with three training functions.

Gradient Descent Algorithms: These algorithms are implemented based on basic gradient descent algorithm or steepest descent algorithm [13] and the weights and biases in the network are updated in the direction of the negative gradient of the performance function. These gradient descent algorithms consist of five BP training methods. They are Gradient Descent (GD), Gradient Descent with Momentum (GDM), Gradient descent with adaptive learning rate (GDA), Gradient descent with momentum and adaptive learning rate (GDX), Resilient back propagation (RP) methods.

Conjugate Gradient Algorithms: In conjugate gradient algorithms, weight adjustments are performed along conjugate directions. Three training methods are Scaled Conjugate Gradient (SCG), Conjugate Gradient back propagation with Fletcher-Reeves Updates (CGF), Conjugate Gradient back propagation with Polak-Riebre (CGP) Updates of BP [14].

Quasi-Newton Algorithms: In these algorithms, the Hessian matrix (second derivatives) can be taken as the current values of the weights and biases. These algorithms converge faster than conjugate gradient methods but with complexity because the computation of Hessian matrix requires more time.

– One Step Secant method (OSS) – The advantage of this algorithm is that storing of Hessian matrix is not required. This algorithm takes the identity matrix as previous Hessian matrix at each and every iteration. It is like a bridge between conjugate gradient and Quasi algorithms.

– Broyden Fletcher Goldfarb Shanno (BFGS) method closer to Newton's method of hill-climbing optimization technique which needs a stationary point of a function. This algorithm requires more storage space and computational time than the conjugate gradient methods but reduce the error effectively.

– Levenberg–Marquardt back propagation (LM) based on the non-linear real-valued functions. The error will be reduced in every iteration and provides accurate results after the completion of all iterations. Due to these features, LM algorithm becomes the fastest and accurate BP method.

LM Algorithm:

It is used to minimize squared error in each iteration by training the ANN model. This error [16] is defined in Eqn. (5).

$$E(W) = \Sigma\left[\Sigma(t_{ij} - o_{ij}0^2\right] \quad (5)$$

Here, $W = [w_1, w_2, ..., w_N]$ all weights of the network, number of weights are denoted by N, t_{ij} is the target output, o_{ij} is computed value from the trained network. Initially weights are fixed and they are updated using the equation given below.

$$W_{K+1} = W_K - (Q^T(R_K).Q(R_K) + lI)^{-1}.Q^T(R_K).E \quad (6)$$

where Q is Jacobian matrix

$$Q(t) = \frac{\partial e(t)}{\partial t} = \begin{bmatrix} \frac{\partial R(P_1, w)}{\partial w_1} & \cdots & \frac{\partial R(P_1, w)}{\partial w_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial R(P_M, w)}{\partial w_1} & & \frac{\partial R(P_M, w)}{\partial w_m} \end{bmatrix} \quad (7)$$

$R(p_i, w)$ is the network function, identity matrix I and a damping factor '*l*'. This method is similar to Gauss-Newton method for *l* = 0. For higher values of 'l', this algorithm is modified to Steepest Decent algorithm. Hence, the value of *'l'* is, automatically adjusted to get secure convergence. The steps involved in LM method [16] are explained in flow chart given in Fig. 4.

In this method, if the learning factor is used to reduce the error in one iteration, then in the next iteration this learning factor is divided by some factor and increase speed to obtain best result. If an increase in error occurs, then this factor was multiplied by another factor to get the proper result. So, with selection of proper values of learning factors, LM method will produce the best results at phase of reconstruction within a short period. Therefore, LM method [17] requires the fewer number of iterations and requirement of memory is also less than other gradient descent and conjugate algorithms.

After training this ANN architecture with different training algorithms explained above, Huffman encoding technique is used at the output of hidden layer to get better compression of image size by reducing the number of bits. This compressed output with weights is applied to output layer and use of activation function to get the reconstructed image at the output.

Quality Metrics: The eminence in medical image compression using neural network is achieved by evaluating the performance metrics given below.

The quality of the image is represented by MSE is mean squared difference between original and decompressed image at the output.

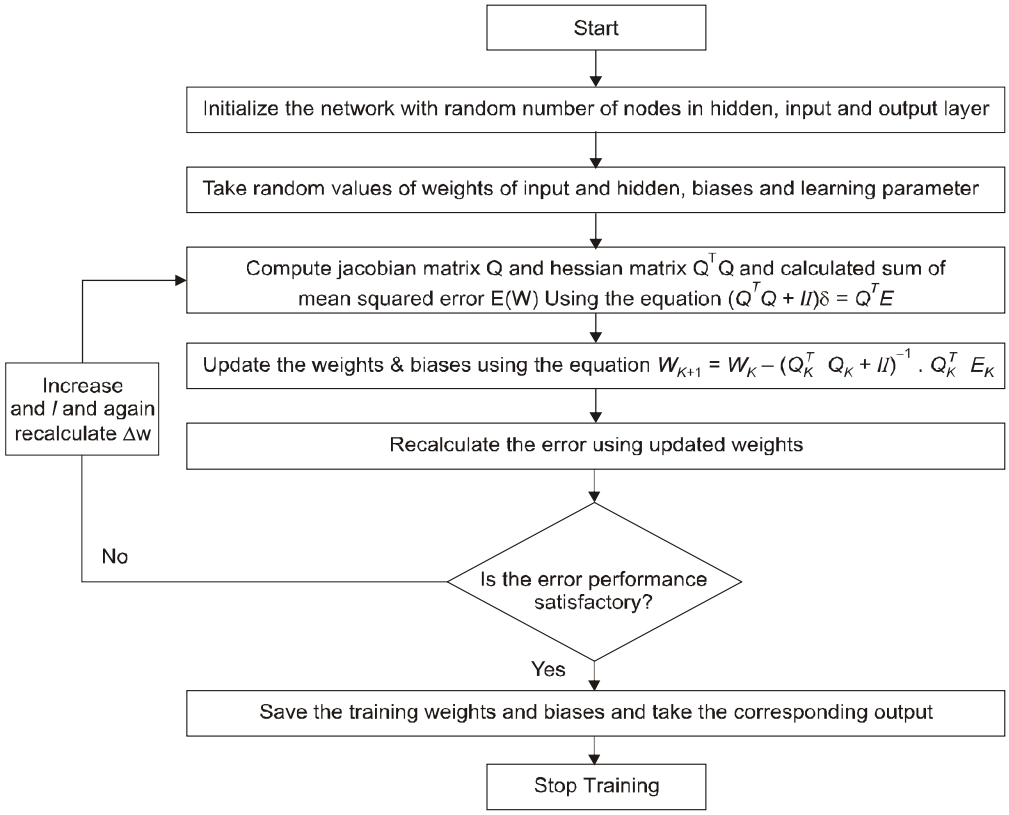

Fig. 4. Flow chart for steps in LM Algorithm.

$$MSE = \frac{1}{XY} \sum_{r=0}^{X-1} \sum_{s=0}^{Y-1} ((A(r,s) - B(r,s))^2) \quad (8)$$

Where $A(r, s)$, $B(r, s)$, denotes the input and decompressed images.

PSNR is the other quality parameter gives the fineness of images. Better quality of output image is reproduced with high value of PSNR and less PSNR gives that less quality of reconstructed image.

$$PSNR(dB) = 10 \log_{10} \left(\frac{255 \times 255}{MSE} \right) \quad (9)$$

Structural Similarity Index Measurement (SSIM) assesses the quality based on the computation luminance, the contrast and the structural part of the image. It gives the perceptual deviation between input and reconstructed images and predicts the quality of decompressed image.

The ratio of input neurons (N_i) to hidden neurons (N_h) is called Compression Ratio (CR) and is defined in Eqn. (10) [18]

$$CR = \frac{N_i}{N_h} \quad (10)$$

III. RESULTS AND DISCUSSION

The experiments have done using FFBPANN model with LM training algorithm to get the reconstructed image at the output. These experiments have done on MRI of brain images collected from Brats data set of brain images. In this paper, FFBPANN model is trained with different training functions of gradient descent, scaled conjugate gradient, LM and their performances are compared with respect to PSNR, MSE, and SSIM. The different training functions are GD, GDM, GDA, GDX, RP of gradient descent; SCG of conjugate gradient, and OSS, BFG, LM of quasi newton algorithms. The network has trained using each training algorithm for 64 inputs at input layer and with hidden neurons of 4, 8, and 16 with the compression ratios of 16:1, 8:1, and 4:1 respectively.

By increasing the no. of hidden nodes from 4 to 16, compression ratio decreased but image quality is enhanced in terms of more PSNR and SSIM. After the completion of training, the evaluated metrics are compared between the training functions and are shown in Table 1, 2 and 3 respectively. Tables 1 to 3 gives evaluated metrics of PSNR, MSE, and SSIM for brain MRI image with block size of 8x8 for the nodes at hidden layer are 4, 8 and 16 respectively. From results of LM training method, it is observed that the average PSNR is better than other algorithms. And the highest PSNR is obtained for 1000 epochs. MSE values of all algorithms are observed and from that results LM algorithm produces less error when compared to others.

From the results, it is also proved that, the LM method gives more SSIM value compared to other algorithms. The encoding technique of Huffman encoding used at hidden nodes to get better compressed output with less number of bits and different timings of encoding, simulation, and training are observed and are given in Table 1, 2 and 3 respectively, for hidden nodes of 4, 8, 16 respectively. The training time for Quasi Newton algorithms of BFG and LM is more compared to other algorithms.

Table 1: PSNR, MSE and SSIM and time for encoding, simulation and training using GD, GDM, GDA, GDX, BFG, RP, OSS, SCG and LM training methods for compression ratio of 16:1.

Training Algorithm	Training methods	PSNR	MSE	SSIM	Time for Encoding (sec)	Time for Simulation (sec)	Training Time (sec)
Gradient Descent	GD	9.9193	6.624e+03	0.0091	0.3963	0.2072	13.9705
	GDM	9.1729	7.8666 e+03	0.0065	0.5440	0.2327	15.6332
	GDA	22.8154	340.048	0.7022	0.5672	0.2188	17.2663
	GDX	23.2928	304.649	0.7165	0.6138	0.2100	17.2815
	RP	23.8199	269.830	0.6863	0.6210	0.2084	18.3927
Conjugate Gradient	SCG	26.4328	147.842	0.8311	0.6099	0.2035	25.4429
Quasi Newton	OSS	25.7518	172.941	0.7623	0.6004	0.1964	32.5603
	BFG	25.3275	190.691	0.7159	0.2940	0.3378	181.9011
	LM	27.8585	106.47	0.8473	0.1411	0.1388	248.4495

Table 2: PSNR, MSE, and SSIM and time for encoding, simulation, and training using GD, GDM, GDA, GDX, BFG, RP, OSS, SCG and LM training methods for compression ratio of 8:1.

Training Algorithm	Training methods	PSNR	MSE	SSIM	Time for Encoding (sec)	Time for Simulation (sec)	Training Time (sec)
Gradient Descent	GD	10.9193	5.2621e+03	0.0192	0.3179	0.2521	15.7181
	GDM	10.1729	6.2487e+03	0.0121	0.3650	0.3723	14.8978
	GDA	25.8154	170.4271	0.7109	0.3354	0.3699	16.7121
	GDX	26.2928	152.6868	0.7262	0.3301	0.4148	15.4216
	RP	26.5199	144.9085	0.7878	0.3152	0.4220	15.6554
Conjugate Gradient	SCG	28.1328	79.3960	0.8716	0.3117	0.4123	17.7714
Quasi Newton	OSS	27.2518	122.4343	0.7979	0.2956	0.4147	26.4321
	BFG	26.3275	151.4704	0.7176	0.6101	0.1985	1231.7
	LM	29.4585	73.6602	0.9087	0.3228	0.0952	2467.7

Table 3: PSNR, MSE, and SSIM and time for encoding, simulation, and training using GD, GDM, GDA, GDX, BFG, RP, OSS, SCG, and LM training methods for compression ratio of 4:1.

Training Algorithm	Training Functions	PSNR	MSE	SSIM	Time for Encoding (sec)	Time for Simulation (sec)	Training Time of network (sec)
Gradient Descent	GD	13.1370	3.1577e+03	0.0257	0.8121	0.4831	34.9045
	GDM	12.5674	3.6003e+03	0.0215	0.3531	0.2124	15.3053
	GDA	25.9971	163.4428	0.6004	0.7541	0.4429	18.2446
	GDX	26.2514	154.1486	0.7053	0.4545	0.2848	17.1513
	RP	27.6033	112.9134	0.8369	0.4614	0.2784	19.0125
Conjugate Gradient	SCG	30.1179	63.2828	0.8917	0.7250	0.4320	32.1491
Quasi Newton	OSS	28.256**0**	97.1575	0.8636	0.7334	0.4465	55.1477
	BFG	27.3982	149.0261	0.8180	0.4112	0.2756	9.6360e +03
	LM	31.7083	43.8788	0.9243	18.7625	15.2886	14515

Table 4: Comparison of Huffman encoded bits at input layer (input image) and hidden layer (Compressed image) with hidden nodes 4, 8, 16, 32 and 64 respectively for LM algorithm.

Hidden Nodes	Huffman encoded bits at input layer (Input Image)	Huffman encoded bits at hidden layer (Compressed output)	Compression ratio
4	65536	4096	16:1
8	65536	8192	8:1
16	65536	16384	4:1
32	65536	32768	2:1
64	65536	65536	1:1

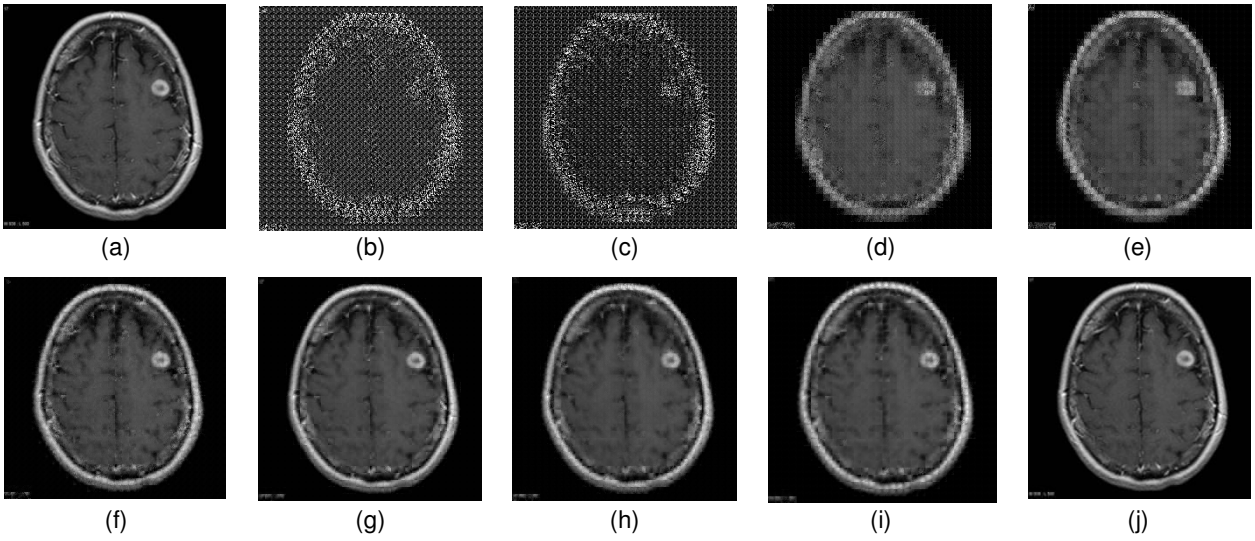

Fig. 5. (a) Brain MRI image (b) reconstructed image of GD (c) reconstructed image of GDM (d) reconstructed image of GDA (e) reconstructed image of GDX (f) reconstructed image of RP (g) reconstructed image of SCG (h) reconstructed image of OSS (i) reconstructed image of BFG (j) reconstructed image of LM training methods with 4:1 compression ratio respectively.

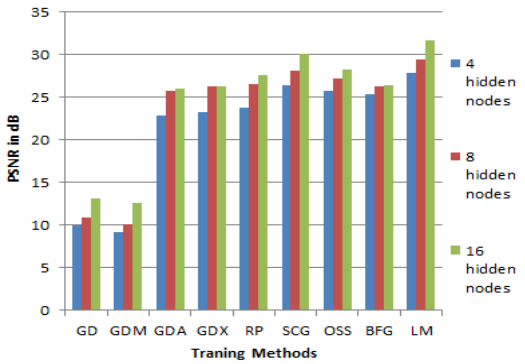

Fig. 6. Graph representing the variation of PSNR values for GD, GDM, GDA, GDX, BFG, RP, OSS, SCG and LM training methods with hidden nodes of 4, 8 and 16 respectively.

From results it is proved that LM takes more time for training compared to other algorithms. By varying the number of nodes at hidden layer (4, 8, 16, 32 and 64), the number of Huffman encoded bits at hidden layer reduces compared with input encoded bits. Hence the compression ratio is also varied.

Table 4 gives these results. Fig. 5 (a)-(j) represents input brain MRI image and reconstructed images of GD, GDM, GDA, GDX, BFG, RP, OSS, SCG, and LM training methods with hidden nodes of 16 respectively.

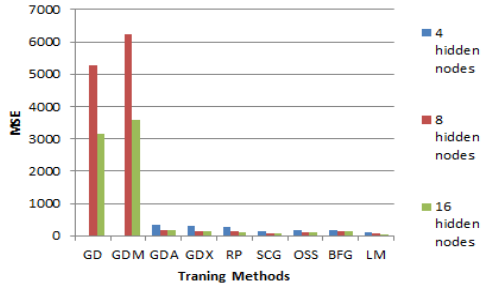

Fig. 7. Graph representing the variation of MSE values for GD, GDM, GDA, GDX, BFG, RP, OSS, SCG, and LM training methods with hidden nodes of 4, 8 and 16 respectively.

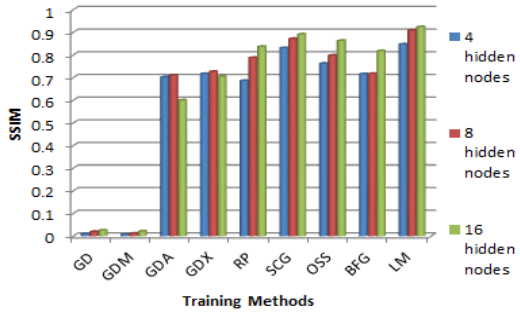

Fig. 8. Variation of SSIM values for GD, GDM, GDA, GDX, BFG, RP, OSS, SCG, and LM training methods with hidden nodes of 4, 8 and 16 respectively.

Fig. 6, 7 and 8 shows the variation in PSNR, MSE, and SSIM values of various training methods GD, GDM, GDA, GDX, BFG, RP, OSS, SCG, and LM of different training algorithms with hidden nodes of 4, 8, 16 respectively.

IV. CONCLUSION

This paper presents FFBPANN model using LM training function with Huffman encoding technique for brain MRI image compression in the region of interest. This network provides better compression ratio, high PSNR, and low MSE. These evaluated metrics of PSNR, MSE, SSIM and encoding, simulation and training time of LM algorithm is compared with other back propagation training methods i.e., GD, GDM, GDA, GDX, BFG, RP, OSS, SCG. From these results, it is concluded that LM training method of Quasi Newton algorithm reveals better medical image compression and reconstruct the original image at the receiver without declining the quality with high PSNR. However, it consumes more time for training as it requires Jacobean functions to reduce error. Finally, it can be concluded that the LM training method of quasi Newton algorithm with Huffman encoding provides better compression compared to gradient descent and conjugate gradient algorithms.

REFERENCES

[1]. Gonzalez, R. C., & Woods, R. E. (2002). Digital Image Processing Second Edition, 2002 by Prentice-Hall.

[2]. Ojha, V. K., Abraham, A., & Snášel, V. (2017). Metaheuristic design of feed forward neural networks: A review of two decades of research. *Engineering Applications of Artificial Intelligence*, *60*, 97-116.

[3]. Arunapriya, B., & Devi, D. K. (2010). Image compression using single layer linear neural networks. *Procedia Computer Science*, *2*, 345-352.

[4]. Gaidhane, V. H., Singh, V., Hote, Y. V., & Kumar, M. (2011). New approaches for image compression using neural network. *Journal of Intelligent Learning Systems and Applications*, *3*(04), 220-229.

[5]. W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu and F. Alsaadi (2017). A survey of deep neural network architectures and their applications. *Neuro Computing*, Vol. *234*,11-26.

[6]. Kumar, V., Sinha, A. K. & Solanki, A. K. (2019). Image Inpainting through Textures Synthesis using Spiking Neural Networks. *International Journal on Emerging Technologies*, *10*(4), 43-49.

[7]. Panda, S. S., Prasad, M. S. R. S., Prasad, M. N. M., & Naidu, C. S. (2012). Image compression using back propagation neural network. *International Journal of Engineering Science and Advance Technology*, *2*(1), 74-78.

[8]. Ahamed, S. A., & Chandrashekarappa, K. (2014). ANN implementation for image compression and decompression using back propagation technique. *International Journal of Science and Research (IJSR)*, *3*(6), 1848-1851.

[9]. Srivastava, R., & Singh, O.P. (2015), Lossless Image Compression Using Neural Network, *International J. of Remote Sensing & Geoscience*, *4*(3), 39-43.

[10] . Jadhav, P., & Siddesh, G. K. (2018). Bandwidth oriented Image Compression using Neural Network with ASAF. *International Journal of Neural Networks and Advanced Applications*, *5*(3), 25-32.

[11]. Tapase, R. (2016). Neural network based image compression. *International Research Journal of Engineering and Technology (IRJET)*, *3*(7), 479-482.

[12]. Raghuwanshi, P., Shrivastava, A. (2017). Improved Image Super Resolution using Sparse codes and Neural Networks. *International Journal of Electrical, Electronics and Computer Engineering*, *6*(1), 51-61.

[13]. Anusha, K., Madhura, G., & Lakshmikantha, S. (2014). Modeling of neural image compression using gradient decent technology. *IJES of Engineering And Science*, *3*(12), 10-17.

[14]. Andrei, N. (2007). Scaled conjugate gradient algorithms for unconstrained optimization. *Computational Optimization and Applications*, *38*(3), 401-416.

[15]. Kumari, S. K. (2017). Implementation and Testing of Image Compression using Neural network. *International Journal of Research and Development in Applied Science and Engineering, 14*(2), 1-7.

[16]. Charles, P. K., Khan, H., Kumar, C. R., Nikhita, N., Roy, S., Harish, V., & Swathi, M. (2010). Artificial Neural Network based Image Compression using Levenberg-Marquardt Algorithm. *International Journal of Modern Engineering Research*, *1*(2), 482-489.

[17]. Gade, M. R. (2014). Image Compression using Multilayer Feed-Forward Artificial Neural Network with Levenberg Marquardt . *International Journal of Engineering and Computer Science, 5*(2), 15681-15684.

[18]. Amirjanov, A., & Dimililer, K. (2019). Image compression system with an optimisation of compression ratio. *IET Image Processing*, *13(*11), 1960-1969.

How to cite this article: Rani, M. Laxmi Prasanna, Rao, G. Sasibhushana and Rao, B. Prabhakara (2020). Medical Image Compression using ANN Model. *International Journal on Emerging Technologies*, *11*(1): 311–318.