Cascade-Forward vs. Function Fitting Neural Network for Improving Image Quality and Learning Time in Image Compression System

Omaima N. Ahmad AL-Allaf

Abstract— **The backpropagation neural network algorithm (BP) was used largely in image and signal processing. The BP requires long time to train the BPNN with small error. Therefore, in this research, three Artificial Neural Networks models (ANNs) were constructed. Three algorithms: FeedForwardNet, CascadeForwardNet and FitNet were adopted to train the three constructed ANNs models separately.**

Each one of constructed models consists of input layer to input the original image, hidden layer to produce the compressed image and finally output layer for decompressed image. The training and testing performance of the constructed models with different architecture were compared to identify the model with best compression ratio (CR) and Peak to Signal to Noise Ratio (PSNR). From experiments, we noted that the better results are obtained when we used the FitNet ANN model. According to results, the performance of constructed FitNet ANN for image compression can be increased by changing the number of hidden layer neurons.

Index Terms— **Image Compression, Artificial Neural Networks, Backpropagation Neural Network, FeedForwardNet, Cascade-ForwardNet, FitNet**

I. INTRODUCTION

Artificial Neural Networks (ANNs) are composed of interconnected neurons that operate in parallel and connected together via weights [1]. ANNs have been used in different applications such as: pattern classification, image and signal processing [2]. The backpropagation neural network (BPNN) is a multi-layer feed forward ANNs. BPNN is useful only when the network architecture is chosen correctly. Too small network cannot learn the problem well, but too large size will lead to over fitting and poor generalization performance [1]. The backpropagation algorithm (BP) can be used to train the BPNN image compression but its drawback is slow convergence. Many approaches have been carried out to improve the speed of convergence [3]. Image compression is a representation of an image with fewer bits to reduce the probability of transmission errors [4]. Many literatures discussed the use of different ANN architectures and training algorithms for image compression to improve the speed of convergence and provide high compression ratio (CR) and high Peak Signal to Noise Ratio (PSNR). Roy et al. (2005) [5] developed an edge preserving image compression technique using one hidden layer feed forward BPNN. Edge detection and multi-level thresholding operations are applied to reduce the image size. The processed image block is fed as single input pattern while single output pattern has been constructed from the original image. Their experiment achieved SNR (0.3013) and CR (30:1) when they applied their approach on Lena image. And, Durai and Saro (2006) [6] suggested mapping the gray levels of the image pixels and their neighbors in such a way that the difference in gray levels of neighbors with the pixel is minimized and then the CR and network convergence can be improved. They achieved that by estimating Cumulative Distribution Function (CDF) for image to map the image pixels. Then, BPNN yields high CR and converges quickly. Their experiments achieved CR (4:1) and PSNR (28.91) when they applied this approach on (256×256) Lena image. And, Rafid (2007) [7] proposed a bipolar sigmoidal BP (PPB) to train a feed forward auto associative NN. Their method includes steps to break down large images into smaller windows for compression process. Experiments have been achieved CR (8:1) and PSNR (29.0) on applying PPB with number of hidden units equal 16 on (256×256) Lena image.

While, Rajapandian and Gunaseeli (2007) [8] proposed modified Backpropagation algorithm (MBP) approach for learning process of BPNN with optimum initialization. The MBP consists of minimizing the sum of squares of linear and non-linear errors for all output units for an efficient process in ANN. They used proper method for weight initialization for good ANN training. And, Xianghong and Yang (2008) [9] used BPNN for image compression and developed algorithm based on BP. The blocks of original image are classified into three classes: background blocks, object blocks and edge blocks, considering the features of intensity change and visual discrimination. Experiments have been achieved: CR (3.156:1) and PSNR (41.209) on applying this approach with number of hidden units (8) on (256×256) Lena image.

Whereas, Fatima (2010) [10] suggested to use multi-layer ANN for image compression. This is done by breaking down large images into smaller windows and applies Discrete Cosine Transform (DCT) to these windows. The input pixels will be used as target values so that assigned mean square error can be obtained.

Omaima N. Ahmad AL-Allaf is currently working in the Department of Computer Information Systems, Faculty of Sciences and Information Technology, AL-Zaytoonah University of Jordan, P.O. Box 130, Amman (11733), Jordan, (Phone: 00962- 4291511 Extension: 347; fax: 00962-4291432; email: omaimaalallaf@zuj.edu.jo).

ISBN: 978-988-19252-1-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

And, Veisi and Jamzad (2009) [11] presented an adaptive BPNN for image compression based on image complexity level by dividing image into blocks, computing complexity of each block and then selecting one network for each block according to its complexity value. They used three complexity measure methods: entropy, activity and pattern-based to determine the level of complexity in image blocks. They used best-SNR approach in selecting compressor network for image blocks which chooses one of the trained networks. Their experiments achieved CR (3.156:1) and PSNR (34.92) on applying the approach with number of hidden units (8) on (256×256) Lena image.

The most important problems that must be solved by researchers are: determining the ANN network architecture, learning parameters and network weights. Therefore, Tai-Hoon et, al. (1991) [12] adopted learning algorithm based on BP algorithm to speed up the learning process by employing the steepness of activation function. This algorithm can converge faster than the BP on some problems but may suffer from increased instability and they frequently fail to converge within a finite time. The cause for the instability is an inappropriate choice for initial weights. To overcome the instability, it is proposed that weight re-initialization be used whenever the convergence speed becomes very slow.

O. AlAllaf (2010) [13] suggested many steps to improve the convergence time for learning the BPNN image compression system. This is done by modifying the BPNN architecture, modifying the BP learning parameters such as learning rate and momentum variable, adding Bais variable, controlling the weights between the layers and so on. Results have been achieved: CR (32:1) and PSNR (44.50) on applying this approach on (256×256) Lena image with 8×8 block dimension and number of hidden neurons equal 2. In another research O. AlAllaf (2011) [14], designed a three layered BPNN for building image compression system. The Fast backpropagation neural network algorithm (FBP) was used for training process to reduce the convergence time. Many techniques were used to improve the use of FBP by: using different BPNN architecture; using different FBP parameters. Finally, FBP results such as CR and PSNR are compared with BP results. From the results, we noticed that the use of FBP improve the BPNN training by reducing the convergence time of learning process.

According to literature studies, we need image compression technique that leads to: less storage requirements; less BPNN training time; and best PSNR and CR. In this research, three Feed Forward ANN architectures were built for image compression system. We adopted three algorithms: FeedForwardNet, CascadeForwardNet and FiNet to train the three architectures. Finally, comparisons between the results of the three algorithms were conducted. The research is organized as follows: section II describes the feed forward ANN architectures. Section III includes details about image compression and decompression system. Section IV describes the results in details. Finally, section V concludes this work.

II. FEED FORWARD ANN ARCHITECTURES

A. FeedForwardNet

Static feed forward ANN has no feedback elements and contains no delays as shown in Fig.1. The output is calculated directly from the input through feed forward connections like BP and Cascade BPNN. The number of connections between each two layers in BPNN is calculated by multiplying the total number of neurons of the two layers, then adding the number of bias neurons connections of the second layer. If there are Ni neurons in input layer, Nh neurons in hidden layer and No neurons in output layer, the total number of connections is given by equation:

Network Size(Nw)= [(Ni×Nh)+Nh]+[(Nh×No)+No] ... (1)

A bias unit is added as a part of every BPNN layer but not the output layer. This unit has a constant value of 1 and it is connected to all units in next layer. The weights on these connections can be trained in the same way as other weights. The bias units provide a constant term in the weighted sum of units in the next layer to improve the convergence time. It contributes a constant term in summation of products (NETj) which is the operand in sigmoid function as shown in Eq.2:

$$NET_j = \sum_{i=1}^{N} X_i W_{ji} + \theta_j \quad \ldots. (2)$$

Momentum variable (α) improves the BP training time and enhancing training stability. It involves adding a term to weight adjustment that is proportional to amount of previous weight change. We used Eq.3 and Eq.4 [3] respectively:

$$W_{ji}^{new} = W_{ji}^{old} + [\Delta W_{ji}^{q}]^{new} \quad \ldots (3)$$

$$[\Delta W_{ji}^{q}]^{new} = \eta \delta_i^q O_j^{q-1} + \alpha [\Delta W_{ji}^{q}]^{old} \quad \ldots (4)$$

Where, α is the momentum variable in the range 0.0-1.0, but it is set to around 0.9. By using momentum, the network tends to follow the bottom of narrow gullies in error surface rather than crossing rapidly from side to side. If α is 0.0, then the smoothing is minimum; the entire weight adjustment comes from the newly calculated change. If α is 1.0, the new adjustment is ignored and previous one is repeated. Between 0 and 1 is a region where weight adjustments are smoothed by amount proportional to α [3]. The variable Beta (β) can be used in sigmoidal function BP to determine the steepness of sigmoid function shape and it lies in range [0.1-1]. When β=0.1, learning is slowly converge, but when β=1, instability may occur. We used Eq.5 and Eq.6 respectively:

$$OUT = F(NET_j) = 1 / (1 + e^{-\beta \times NET_j}) \quad \ldots\ldots.. (5)$$

$$F'(NET_j) = \beta \times (OUT\ (1 - OUT)) \quad \ldots\ldots\ldots (6)$$

B. CascadeForwardNet

These are similar to feed forward networks such as BPNN with the exception that they have a weight connection from the input and every previous layer to the following layers. Fig.2 shows a three-layer network has connections from layer 1 to layer 2, layer 2 to layer 3, and layer 1 to layer 3. The three-layer network also has connections from the input to all three layers [15],[16].

C. FitNet

The function fitting neural networks are also a type of feed forward networks, which are used to fit an input output

ISBN: 978-988-19252-1-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

relationship. A feed forward network with one hidden layer and enough neurons in the hidden layers can fit any finite input-output mapping problem [15][16].

III. Image Compression/Decompression System

The design of feed forward ANN for image compression system involves determining the number of network's layers. In this research, we used ANN architecture with three layers. Three feed forward ANN models (FeedForwardNet, CascadeForwardNet and FitNet) were constructed for image compression to obtain good results for CR and PSNR and reducing the training time. Fig.1 shows the FeedForwardNet and FitNet image compression/decompression system whereas Fig.2 shows the CascadeForwardNet image compression/decompression system. The Feed forward ANN learning process on image compression requires input, hidden and output layers of ANN. The learning process is adopted using three ANN algorithms (FeedForwardNet, Cascade-Forward and FitNet) using a set of images as training patterns. After finishing the learning process, the Feed forward ANN image compression process requires the input and hidden layers. But the image decompression process requires the hidden and output layers.

The next step in designing the BPNN involves determining the number of neurons in each layer. This ANN is fed by a colored 256×256 image as an input and produces a compressed code at the hidden layer units. In the reconstruction process, this ANN model produces a colored 256×256 image by output layer units. The number of neurons in input layer (N_i) is equal to the number of neurons in output layer (N_o). The Ni depends on the dimension of image sub block (P×P).

The input and output layers have Ni units each, and an intermediate layer with N_h units where N_h is less than N_i. The number of hidden layer units (N_h) is less than the number of input layer units. The number of hidden layer units effects on compression performance of BPNN.

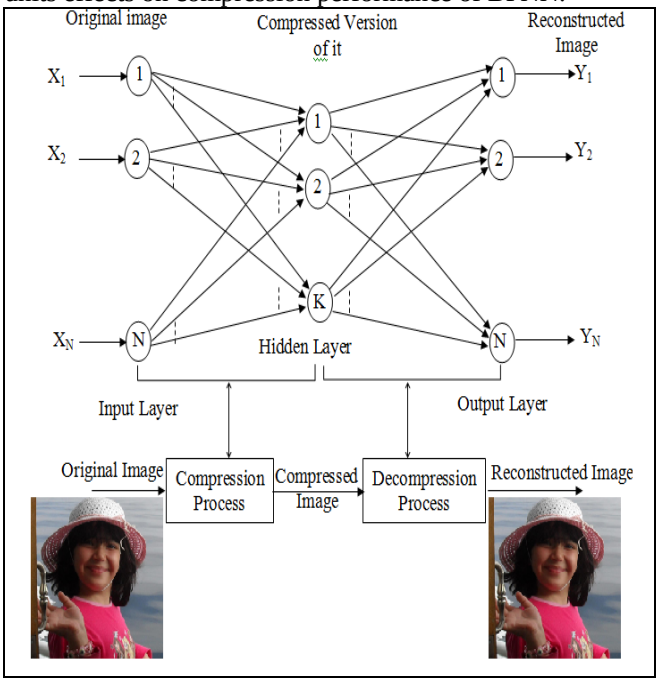

Fig. 1. FeedForwardNet Image Compression [13]

In this research, we suggested to use the same processes which adopted in our previous research [13] such as: image normalization and segmentation; initialization BPNN learning parameters, weight connections); and preparation of training and testing set.

A. FeedForwardNet Simulation Program

The simulation program of each one of conducted training algorithm (FeedForwardNet, CascadeForwardNet and FitNet) includes the following steps:

1) Initialization of network weights, learning rate (η) and Threshold error. Set iterations to zero.
2) Open the file which contains the image training set.
3) Total_error = zero; iterations → iterations+1
4) Get one vector from file and feed it to input layer units.
5) Initialize the target output of that vector.
6) Calculate the outputs of hidden layer units.
7) Calculate the outputs of output layer units.
8) Calculate the error = desired output – actual output Total_error → Total_error + error
9) Calculate delta sigma of output neurons. Then adjust weights between output and hidden layer units.
10) Calculate delta sigma of hidden layer units. Then adjust weights between hidden and input layer units.
11) While there are more vectors in the file, go to step 4.
12) if Threshold error >= Total_error then stop, otherwise go to step 3.

B. FeedForwardNet Compression Process

The compression process includes the following steps:

Step 1: Read image pixels from file.
Step 2: Divide the image into non-overlapping blocks.
Step 3: Apply the image block into input layer units.
Step 5: Compute the outputs of hidden layer units by multiplying the input vector by weight matrix (V).
Step 6: Store hidden layer outputs in a compressed file.
Step 7: While there are more image vectors go to step4.

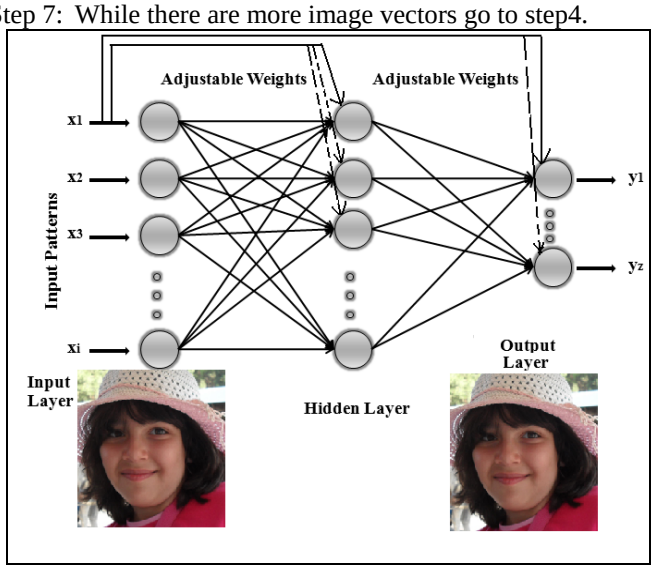

Fig. 2. Cascade-forward ANN

C. FeedForwardNet Decompression Process

The decompression process includes the following steps:

Step 1: Open the compressed file.
Step 2: Take one vector from the file.

ISBN: 978-988-19252-1-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Step 3: Compute outputs of output neurons by multiplying outputs of hidden layer units by weight matrix (W).
Step 5: Take the outputs of output layer units (sub image of size P×P) and put it in its proper location in the reconstructed file.
Step 8: While there are more vectors in compressed file go to step2.

IV. Experimental Results

In this research, three simulation programs to implement FeedForwardNet, CascadeForwardNet and FitNet algorithms were written using MathLab software. These programs include the implementation of steps of ANN compression and decompression system. In this research, we used 15 colored (256×256=65536) images as shown in Fig.3 as training set for learning process. We used block dimension equal (8×8=64) to segment each image into many blocks. By dividing image size (65536) on the block dimension (64), we got number of blocks 1024 for each image. As a result the total number of training blocks of all images is equal to 15360 (1024×15) to train 15 images.

To check the compression performance, the compression ratio (CR) and bit per pixel rate (bpp) are calculated. The CR is the degree of data reduction obtained as a result of compression process, whereas bpp is the number of bits required to represent each pixel value of compressed image.

In ANN image compression system, the CR is defined by the ratio of data fed to the input layer Neurons (Ni) to the data out from the hidden layer neurons (Nh). The CR can be expressed as Eq.7

$$CR = \frac{Ni \times (\text{pixel size in bits})}{Nh \times (\text{pixel size in bits})} \quad \ldots\ldots\ldots \quad (7)$$

Also the CR can be computed by the Eq.8:

$$CR = (1 - (Nh/Ni)) \times 100\% \quad \ldots\ldots\ldots\ldots \quad (8)$$

After completing the decoding process SNR, PSNR and Normalized Mean Squared Error (NMSE) must be calculated between the reconstructed image and original image to verify the quality of decoded image. The better compression performance is with the highest CR, the least bpp rate and highest PSNR [4]. Simulations were conducted to evaluate the compression and generalization performances of the conducted ANN image compression system. The efficiency of this system was tested by several experiments using real world images.

A. The Effect of Hidden Layer Neurons

The ANN was trained using FeedForwardNet on 15 images (D01, D02,… and D15) with dimension (256×256) with different numbers of hidden units: 10, 15, 24, 34, 44, 54 and 64 respectively.

Table I shows the impact of number of hidden layer units on number of training iterations, CR and Bpp. The best value of CR is obtained when ANN is trained and tested with 10 hidden numbers of units.

TABLE I
THE EFFECT OF HIDDEN LAYER NEURONS ON CR

no. of hidden units	no. of iterations	CR	BPP
10	1320	84.37	0.1563
15	1231	76.56	0.2344
24	976	62.5	0.3750
34	931	46.87	0.5313
44	885	31.25	0.6875
54	893	15.62	0.8438
64	854	0	1

At the same time, Table II shows the impact of number of hidden layer units on RMSE, SNR and PSNR when ANN was trained using FeedForwaredNet.

TABLE II
IMPACT OF HIDDEN LAYER NEURONS ON PSNR

No. of hidden units	RMSE	SNR	PSNR
10	4.3216	36.234	42.3521
15	3.7652	31.3256	37.5089
24	5.4165	29.4321	36.4322
34	4.3065	28.1879	33.08245
44	4.5643	29.4567	35.4356
54	3.5431	31.3451	35.16616
64	3.7652	32.6834	39.4482

B. Comparisons between Cascaded and FitNet

The feed forward ANN model is trained using (FeedForwardNet, Cascade-Forward and FitNet) algorithms separately many times with different number of hidden layer neurons (Nh) such as 10, 15, 24, 34, 44, 54 and 64 respectively. The trained ANN is tested with one of the trained images (D01), untrained images (D11) and (D16) shown in Fig.4. Table III shows the results of three algorithms according to number of iterations, SNR and PSNR and RMSE. Because we use the same Ni=64 and Nh=8, the CR is same for the three algorithms and it is equal 0.1563. Also Bpp is same and equal 84.3750. We note from Table III that FitNet algorithm has the best results for PSNR. Fig.5 shows the original trained image (D01) and reconstructed images after decompressing them with different Nh when ANN is trained using FitNet algorithm.

ISBN: 978-988-19252-1-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Fig. 3. The 15 images (256x256) which used for Feed Forward ANN training


Fig. 4. Untrained Images (D11) and (D16) used in ANN Testing

Table IV shows the results of three algorithms according to number of iterations, SNR and PSNR and RMSE. Because Ni=64 and Nh=15, the CR is same for the three algorithms and it is equal 0.2344. Also the Bpp is same and it is equal 76.5625.

TABLE III
DIFFERENCES BETWEEN THREE MODELS (NH=8)

Model	img	Iterations	SNR	PSNR	RMSE
feedforward net	D01	1320	20.43	24.282	19.60
	D11	Test	21.28	25.137	10.72
	D16	Test	21.23	21.035	17.11
Cascade-Forward	D01	1202	22.65	27.542	15.97
	D11	Test	25.66	27.637	9.723
	D16	Test	22.87	23.874	15.97
FitNet	D01	1103	25.76	28.876	11.76
	D11	Test	26.22	28.866	9.000
	D16	Test	23.98	24.873	14.87

TABLE IV
DIFFERENCES BETWEEN THREE ANN MODELS (NH=15)

Model	img	Iterations	SNR	PSNR	RMSE
feedforward net	D01	1231	23.65	28.883	14.53
	D11	Test	24.22	27.228	9.232
	D16	Test	24.43	24.463	14.06
Cascade-Forward	D01	1112	27.54	30.54	12.76
	D11	Test	25.00	28.541	8.171
	D16	Test	24.99	25.824	13.98
FitNet	D01	998	29.98	31.651	10.76
	D11	Test	26.11	29.783	7.998
	D16	Test	25.76	27.858	11.83

We note from Table IV that the FitNet algorithm has the best results for PSNR. Table V shows the results of three algorithms according to number of iterations, SNR and PSNR and RMSE. Because (Ni=64 and Nh=24), the CR is same for the three algorithms and it is equal 0.3750. Also the Bpp is same and it is equal 62.5.

TABLE V
DIFFERENCES BETWEEN THREE ANN MODELS (NH=24)

Model	img	Iterations	SNR	PSNR	RMSE
feedforward net	D01	976	24.00	31.2558	11.05
	D11	Test	27.29	30.8732	7.228
	D16	Test	25.00	26.1139	12.61
Cascade-Forward	D01	892	32.87	34.7724	8.762
	D11	Test	29.22	31.2764	7,005
	D16	Test	25.83	28.8373	10.27
FitNet	D01	789	31.87	36.8273	7.882
	D11	Test	30.20	32.2732	6.743
	D16	Test	26.29	30.3737	10.00

We note from Table V that the FitNet algorithm has the best results for PSNR and it needs less time for training process. Table VI shows results of the three algorithms according to number of iterations, SNR and PSNR and RMSE. Because (Ni=64 and Nh=34), the CR is same for the three algorithms and it is equal 0.5313. Also the Bpp is same and it is equal 46.8750. We note from Table VI that the FitNet algorithm has the best results for PSNR.

TABLE VI
DIFFERENCES BETWEEN THREE ANN MODELS (NH=34)

Model	img	Iterations	SNR	PSNR	RMSE
feedforward net	D01	931	29.12	32.5190	9.562
	D11	Test	29.27	31.1311	6.343
	D16	Test	27.09	28.5164	10.73
Cascade-Forward	D01	821	32.98	35.8755	8.005
	D11	Test	30.39	33.0232	6.000
	D16	Test	28.75	30.8458	9.757
FitNet	D01	716	34.98	36.9998	7.988
	D11	Test	32.29	35.2829	5.933
	D16	Test	29.93	34.3555	8.845

Table VII shows the differences between the three algorithms according to number of iterations, SNR and PSNR and RMSE. Because Ni=64 and Nh=44), CR equal 0.6875 and Bpp is equal 31.2500. We note from Table VII that the FitNet algorithm has the best results for PSNR and needs less time for training process.

ISBN: 978-988-19252-1-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

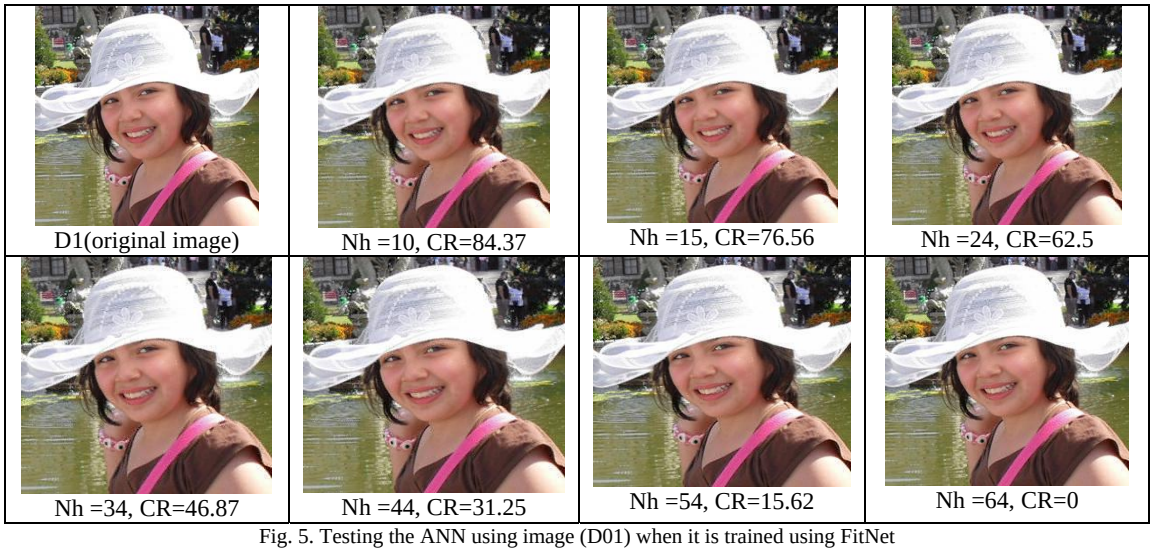

Fig. 5. Testing the ANN using image (D01) when it is trained using FitNet

TABLE VII
DIFFERENCES BETWEEN THREE ANN MODELS (NH=44)

Model	img	Iterations	SNR	PSNR	RMSE
feedforward net	D01	885	29.99	33.0388	9.007
	D11	Test	30.77	32.6262	6.000
	D16	Test	26.10	27.9818	9.172
Cascade-Forward	D01	789	32.87	36.9809	7.987
	D11	Test	32.79	35.6388	5.282
	D16	Test	26.00	32.8723	8.096
FitNet	D01	694	32.00	37.0087	6.987
	D11	Test	33.87	37.3855	5.002
	D16	Test	27.93	33.2302	7.000

Finally, Table VIII shows the results between the three algorithms according to number of iterations, SNR and PSNR and RMSE. Because Ni=64 and Nh=54, the CR is same for is equal 0.8438. Also the Bpp is same and it is equal 15.6250. We note from Table VIII that the FitNet algorithm has the best results for PSNR.

TABLE VIII
DIFFERENCES BETWEEN THREE ANN MODELS (NH=54)

Model	img	Iterations	SNR	PSNR	RMSE
feedforward net	D01	893	31.45	37.9342	8.125
	D11	Test	31.37	33.2332	5.139
	D16	Test	27.12	28.9970	8.050
Cascade-Forward	D01	791	32.98	37.9987	7.876
	D11	Test	33.37	37.4342	5.006
	D16	Test	28.76	31.7651	7.987
FitNet	D01	699	33.00	38.0098	6.097
	D11	Test	34.82	39.2882	5.000
	D16	Test	28.83	32.8842	7.000

From Tables (III, IV, V, VI, VII and VIII) we can note that the training time of FitNet is less than the training time of FeedForwardNet and cascade-forward.

V. Conclusion

In this research, three feed forward ANN models were constructed for image compression system. The ANN architecture for the three models was consisted of input, hidden and output layers. The FeedForwardNet, CascadeForward and FitNet algorithms were adopted separately to train the constructed ANN models. Different ANN architectures were used for the three ANN models. This is done by changing the number of hidden layer neurons to increase compression performance (Bpp and CR). We used 15 colored (256×256) images for the training process. Experiments were conducted to check the performance of each algorithm. The experiments were based on both trained and untrained images to check the generalization of feed forward ANN.

The PSNR results obtained by testing the feed forward ANN which was trained using Fitnet are better than the PSNR results obtained from ANN which was trained using FeedForwardNet and cascadForward. At the same time, the ANN architecture required less training time for image compression\decompression system when we used FitNet training algorithm than when we used FeedForwardNet and Cascade-Forward. From results, we note that the FitNet algorithm can be successfully reduced the learning time (convergence time) in comparison with the FeedForwardNet and Cascade-Forward. At the same time with maintaining the reconstructed image performance (CR and PSNR). For future work, we suggest using other ANN architecture then we will make comparison with FitNet.

ACKNOWLEDGEMENT

The author would like to thank AL-Zaytoonah University of Jordan for supporting this research.

REFERENCES

[1] R. P. Lippmann. "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, vol.4, no.2, April 1987, pp.4-22.

[2] N. K. Ibrahim, R.S.A. Raja Abdullah and M.I. Saripan. "Artificial Neural Network Approach in Radar Target Classification," *Journal of Computer Science*, vol. 5, no.1, 2009, pp.23-32, ISSN: 1549-3636, Science Publications.

[3] P. D. Wasserman. *Neural Computing: Theory and Practice*, Van Nostrand Reinhold Co. New York, NY, USA, 1989, ISBN: 0-442-20743-3.

[4] R. C. Gonzales and P. Wintz. *Digital Image Processing*, second edition, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1987, ISBN: 0-201-11026-1.

[5] Senjuti B.Roy, K. Kausik and S. Jaya, 2005. Edge Preserving Image Compression Technique Using Adaptive Feed Forward Neural Network, *Proc. of the 9th IASTED International conference on internet and multimedia systems and applications*, Grindelwald, Switzerland, Feb 2005, pp. 467-471.

[6] Durai S. A. and E .A. Saro, 2006. Image Compression with Back-propagation Neural Network Using Cumulative Distribution Function, *World Academy of Science, Engineering and Technology*, vol.17, pp. 60-64.

[7] Rafid A. K., 2007. Digital Image Compression Enhancement Using Bipolar Backpropagation Neural Networks, *Al-Rafidain Engineering*, 15(4):40-52.

[8] V.V. Joseph Rajapandian and N. Gunaseeli. "Modified Standard Backpropagation Algorithm with Optimum Initialization for Feedforward Neural Networks," *International Journal of Imaging Science and Engineering (IJISE)*, vol.1, no.3, July 2007, GA, USA, ISSN: 1934-9955.

[9] T. Xianghong and L. Yang. "An Image Compressing Algorithm Based on Classified Blocks with BP Neural Networks," *Proc. of the international conference on computer science and software engineering*, IEEE Computer Society, Wuhan, Hubei, vol. 4, Dec 2008, pp.819-822, DOI: 10.1109/CSSE.2008.1357.

[10] Fatima B. Ibrahim. "Image Compression using Multilayer Feed Forward Artificial Neural Network and DCT," *Journal of Applied Sciences Research*, vol.6, no.10, 2010, pp. 1554-1560.

[11] Veisi H. and M. Jamzad, 2009. A Complexity-Based Approach in Image Compression Using Neural Networks, *International Journal of Signal Processing*, 5(2): 82-92, ISSN: 2070-397X,

[12] C. Tai-Hoon, R. W. Conners and P. A. Araman. "Fast Back-Propagation Learning Using Steep Activation Functions and Automatic Weight Reinitialization," Conference Proceedings, 1991 *IEEE International Conference on Systems, Man, and Cybernetics* "*Decision Aiding for Complex Systems*", Omni Charlottesville Hotel and University of Virginia, Charlottesville, Virginia, Vol. 3, 13-16October1991, pp.1587-1592.

[13] Omaima N. A. AL-Allaf. "Improving the Performance of Backpropagation Neural Network Algorithm for Image Compression/Decompression System," *Journal of Computer Science*, DOI: 10.3844/jcssp.2010.1347.1354, vol.6, Issue.11, 2010, pp. 1347-1354.

[14] Omaima N. A. AL-Allaf, Fast BackPropagation Neural Network Algorithm for Reducing Convergence Time of BPNN Image Compression, The 5th International conference on Information Technology and Multimedia (ICIMμ2011), November 14-16, 2011, Kuala Lumpur, Malaysia.

[15] O. De Jesus and M. T. Hagan, "Backpropagation Algorithms for a Broad Class of Dynamic Networks," *IEEE Transactions on Neural Networks*, vol.18, no.1, pp.14 -27, Jan.2007.

[16] MathWorks, Neural Network Toolbox 7.0, MathWorks Announces Release 2010a of the MATLAB and Simulink Product Families, 2010, MathWorks, Inc. www.mathworks.com/trademarks.

ISBN: 978-988-19252-1-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)