www.ietdl.org

Published in IET Image Processing
Received on 19th February 2011
Revised on 5th January 2012
doi: 10.1049/iet-ipr.2011.0079

ISSN 1751-9659

Bi-level image compression technique using neural networks

S. Sahami[1] *M.G. Shayesteh*[1,2]

[1]*Department of Electrical Engineering, Urmia University, Urmia, Iran*
[2]*Wireless Research Laboratory, ACRI, Electrical Engineering Department, Sharif University of Technology, Tehran, Iran*
E-mail: m.shayesteh@urmia.ac.ir

Abstract: This study presents the utilisation of neural-network for bi-level image compression. In the proposed lossy compression method, the locations of pixels of image are applied to the inputs of a multilayer perceptron neural-network. The output of the network denotes the pixel intensity (0 or 1). The final weights of the trained neural-network are quantised, represented by a few bits, Huffman encoded and then stored as the compressed image. In the decompression phase, by applying the pixels locations to the trained network, the output determines the intensity. The results of experiments on more than 4000 different images indicate higher compression rate of the proposed structure compared with the commonly used methods such as Comité Consultatif International Téléphonique of Télégraphique (CCITT) G4 and joint bi-level image experts group (JBIG2) standards. Moreover, quantisation issue in neural-network deployment is addressed and a solution is proposed. Further, an adaptive technique based on binary image characteristics is applied to achieve more compression rates.

1 Introduction

Data compression is an active field of research and has received great attention in the business and academic research. Although demands are always rising for higher data rates and better deployment of data processing, the efforts for compressing data seem endless. Image, as an information source, can be transmitted via different representations. The fewer bits required for representation, the more compression ratio (CR) could be achieved. There are two different paradigms for image compression: lossy and lossless [1]. In the lossless approach, no information loss would be tolerated. However in the lossy scheme, some less relevant information is sacrificed in order to obtain higher compression rates.

Shannon [2] derived a bound on information representation which determines the minimum possible number of bits for representation of a set of symbols. Compression ratios in lossless approaches are limited by Shannon bound, whereas their lossy counterparts can achieve higher rates. Since image information is judged by psychovisual aspect of human perception, Jayant [3] proposed a bound for image compression subjectively. This bound is looser than that of Shannon.

So far, different approaches for image compression have been proposed. These methods can be categorised into signal processing and artificial intelligent-based methods, for example fractal [4, 5], vector quantisation [6], dimension reduction [7] and domain transform [8].

Several artificial intelligence methods for image compression have been investigated. Among them, genetic algorithms (GA) and neural networks (NN) have been widely used [4, 9–13]. GA-based methods are usually used as an optimisation technique for fractal image compression [4, 9, 10] or as clustering approaches such as vector quantisation [11]. GA has also been used for binary image compression [12]. However, GA-based approaches are limited in many aspects and have not received considerable attention as neural networks. In fact, compression techniques based on GA need much more computational time and complexity in comparison with the neural-network-based methods and do not result in high-quality reconstructed images [12].

Neural-network approach has been used for grey scale and colour image compression in [7] and [13–15]. In the methods applied on grey-level images, each image is divided into sub-blocks and each sub-block is applied to a multilayer perceptron (MLP) neural-network. MLP is used to find a mapping that represents the sub-blocks with a lower number of outputs. However, the network cannot find the mapping for all sub-blocks of image correctly. Therefore sub-blocks are first mapped to higher dimensions, and then the network is trained in order to find a mapping for lower representing neurons. To train the neural-network, the decoder is cascaded directly after the encoder. Then, the network is trained in a way that the output estimates the input image. The encoder output layer represents the encoded sub-block. One sub-block is fed to network for training and this will be repeated for other sub-blocks as well. The compressed image should provide two categories for the decoder: first, the output of encoding part for each sub-block should be included in the compressed file. Second, the weights of decoding part of network must be also sent in the compressed image since the receiver needs a specific network for each image to reconstruct the original one.

However, these methods are not capable of achieving high data compression ratios [13].

Binary images can be divided into three regions: text, halftone and generic (such as shape). There are different methods for compression of distinct parts of a bi-level image [16–26]. For example, the recent studies [16, 17] use chain codes and achieve slightly more compression ratio than the JBIG2 standard which is the best lossless standard in this field [18–21]. In [26], binary compression has been performed by applying a context-weighting algorithm where the performance is slightly better than JBIG1 [1]. Some other techniques have been also suggested for binary image compression such as logic spectra or automata-based methods [27, 28].

In this paper, we deal with the generic part of binary document. Generic elements usually can be considered as a collection of several shape elements. Shape coding techniques can be categorised into two main types: bitmap and contour-based techniques. In bitmap scheme, a pixel-to-pixel approach is considered, whereas in the second one, the emphasis is on the coded representation of shapes borders [16, 17].

In this study, we present another paradigm of neural-network deployment for binary image compression. In our approach, a neural-network is trained to classify the inputs (which are the locations of pixels) into two classes 0 or 1. Zero and one indicate that the given address belongs to background and foreground, respectively. We use multilayer perceptron with one hidden layer for this purpose. The desired output (target) of the network represents the pixel intensity which is 1 for white and 0 for black. After the network is trained, the final weights are used instead of original binary image (compressed image). At the reconstruction phase, the pixels addresses are fed to the network inputs and the network output is obtained. Comparing the output with the threshold 0.5, the intensity value 0 or 1 for that pixel is assigned. We evaluate the proposed method on two databases which include more than 4000 different images with diverse type from simple to complex shapes. The experimental results indicate that our technique achieves high compression rate as well as high peak signal to noise ratio (PSNR). Moreover, we discuss about the quantisation issue in the neural-network deployment. We also apply Huffman encoding to the weights to achieve more compression ratios. Further, activity, pattern-based criteria and some other binary image features are used in an adaptive algorithm for better utilisation of the network.

This paper is organised as follows. In Section 2, the proposed method is explained. Section 3 provides experimental results and compares them with several common methods. Finally, conclusion is presented in Section 4.

2 Proposed method

2.1 Architecture

Fig. 1 demonstrates block diagram of the proposed method in the compression phase. As shown, a multi-layer perception neural network with one hidden layer is employed. The neural network has two inputs: pixel locations (x, y). The desired output (target) of the network is the intensity of that pixel. The performance measure criterion in the network training is the mean square error (MSE). That is, the network is trained in a way that the MSE between the target (desired output; pixel intensity) and the network output is

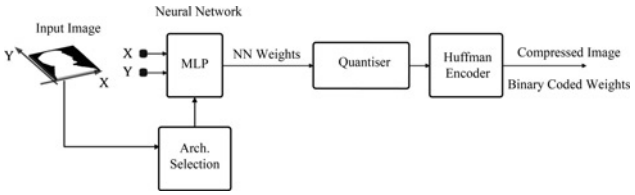

Fig. 1 *Block diagram of the proposed method (compression phase)*

minimised, that is, the output will be as close as possible to the intensity of the pixel. Levenberg–Marquardt (LM) back propagation algorithm is used for training. After training, the final weights of the network are quantised and then represented by binary numbers. In order to achieve higher compression rates, we use an efficient coding technique such as Huffman coding [29]. The outputs of Huffman encoder are stored instead of intensities. Since, the number of weights (which depends on the number of neurons) and consequently their encoded binary representations are much less than the number of pixels; hence we achieve high compression rates.

Image characteristics such as activity, pattern-based complexity and several other features are used to determine the number of neurons adaptively.

The neural-network classifies the input address into two different classes. One class denotes pixels with zero intensity (black) and the other one represents white pixels. The binary characteristic of image provides the opportunity to apply the new method effectively in image compression. We just use the weights of the network as the compressed image, whereas as mentioned, the previously presented neural-network-based methods need an additional set of inputs for decompression. Since in the new method, the inputs are the pixels locations which are the same for all images; hence it is not necessary to consider any further information. Therefore the new scheme can reach the lower level of information rate per pixel in contrast to the previous methods which do not achieve such compression rates.

Fig. 2 demonstrates the reconstruction steps of the new method. As observed, the binary coded weights are decoded using Huffman decoder and then by applying the pixels locations to the inputs of the trained network, the output is computed. Comparing the network output with the threshold value 0.5, the binary output (pixel intensity) is determined, that is, the image is reconstructed. In order to enhance the image, some post-processing techniques such as morphological approaches can be applied. However, we did not use these operations in this work.

Here, each image has its own specific neural-network which differs from those of the other images. Each specific neural-network represents only one image. So in the

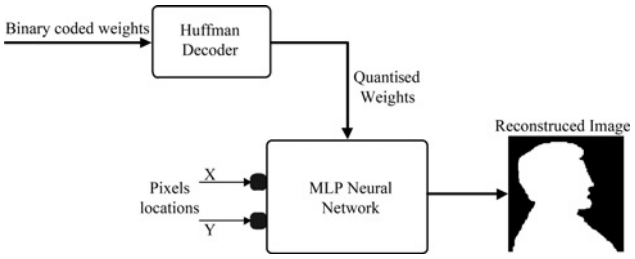

Fig. 2 *Block diagram of the proposed method (decompression phase)*

compression phase, we train one network with one image and in the decompression phase, we use the trained network to reconstruct the corresponding image. In this study, we consider the neural-network classifier as a decoder which decodes the image pixel intensities based on the weights obtained during the training process. Encoder and decoder are the same in our application.

2.2 Quantisation

Noting Fig. 1, after training the network the final weights are achieved. The weights are real numbers and should be represented by binary digits. Therefore quantisation is necessary which affects the performance of trained neural-network in the reconstruction phase. On the one hand, the weights should be represented by sufficiently large number of bits in order to decompress high-quality image. On the other hand, large number of bits decreases the compression ratio. Therefore there is a compromise between the quality and compression ratio which is the bottleneck of the technique. Consequently, an efficient solution should be applied for weights quantisation. In the subsequent paragraphs, we analyse the effects of quantisation and number of assigned bits on the neural network performance.

In Fig. 3, the distribution of the weights of 4173 different trained networks corresponding to 4173 images has been shown. It is observed that the interval $[-32\ 32]$ spans most of the weights. Our experimental investigation shows that the out of band weights mostly belong to the non-converged networks (near 1.06% of trained network). Further, about 90% of the weights lie in the interval $[-20\ 20]$.

Noting weights distribution, we use non-uniform quantisation which shows better results than the uniform quantisation. Let B be the number of bits assigned to each weight, then the number of different levels will be 2^B. Each weight lying in a specific interval is quantised to the nearest lower or upper limit of that interval.

The distribution of the quantisation error which is also called round-off noise has been depicted in Fig. 4 for different number of representing bits (B). It is observed that the error is proportional to B [30]; that is, the more representing bits, the less rounding error would impact the network.

To analyse the effect of round-off noise on the performance, we note that the network output (z) without weight quantisation is computed as

$$z = f_{\text{out}}(\boldsymbol{W}_2 \times f_{\text{hid}}(\boldsymbol{W}_1 \times \boldsymbol{P})) \tag{1}$$

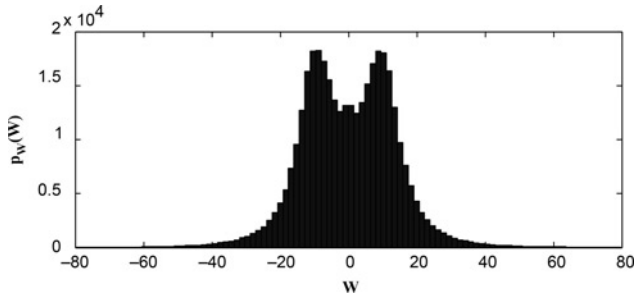

Fig. 3 *Distribution of weights of 4173 trained neural networks*

where $\boldsymbol{W}_1 = [w_{1,ij}]_{H\times 2}$ is the first layer weight matrix, H is the number of hidden layer neurons, $\boldsymbol{W}_2 = [w_{2,i}]_{1\times H}$ is an H element vector that represents the weights of the second layer and $\boldsymbol{P} = \begin{bmatrix} x & y \end{bmatrix}^{\mathrm{T}}$ is the input vector. f_{out} and f_{hid} are the output and hidden layer activation functions, respectively. After weight quantisation ($\boldsymbol{W}_i^q$), the output, z_q, can be written as

$$\begin{aligned} z_q &= f_{\text{out}}((\boldsymbol{W}_2^q) \times f_{\text{hid}}((\boldsymbol{W}_1^q) \times \boldsymbol{P})) \\ &= f_{\text{out}}((\boldsymbol{W}_2 + \boldsymbol{V}_2) \times f_{\text{hid}}((\boldsymbol{W}_1 + \boldsymbol{V}_1) \times \boldsymbol{P})) \end{aligned} \tag{2}$$

where $\boldsymbol{V}_i$ is the quantisation noise matrix (vector). Each element of $\boldsymbol{V}_i$ is a sample of the distribution depicted in Fig. 4. The difference between the network outputs owing to the quantised and non-quantised weights will be as

$$\varepsilon = z_i - z_q \tag{3}$$

The activation function used in this study is tangent sigmoid function $f(x) = [2/(1 + \mathrm{e}^{-2x}) - 1]$. It can be approximated as: $f(x) \cong \begin{cases} x & |x| \le 1 \\ 1 & |x| > 1 \end{cases}$. Therefore z_q can be written as (see (4))

where $I = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^{\mathrm{T}}$ is an H element vector. The above statement can be also applied to z.

In the first case of (4), we have $z \simeq 1$ and $z_q \simeq 1$; hence $\varepsilon = 0$ which means quantisation does not affect the output.

In the second case, we obtain $\varepsilon = \boldsymbol{V}_2 \times \boldsymbol{I} = \sum_{i=1}^{H} v_{2,i}$. As can be seen from Fig. 4, each element of quantisation vector is very small. Further, our experiments show that the number of hidden layer neurons to achieve good performance is less than 50. Therefore it can be concluded that ε cannot be larger than the noise margin and change the output (note that we consider the network output as '1' when the actual output is more than 0.5 and '0' otherwise).

In the third case which is crucial, we have

$$z = x\sum_{i=1}^{H} w_{2,i}w_{1,i1} + y\sum_{i=1}^{H} w_{2,i}w_{1,i2} \tag{5}$$

$$\begin{aligned} z_q &= x\sum_{i=1}^{H} (w_{2,i} + v_{2,i})(w_{1,i1} + v_{1,i1}) \\ &\quad + y\sum_{i=1}^{H} (w_{2,i} + v_{2,i})(w_{1,i2} + v_{1,i2}) \end{aligned} \tag{6}$$

The error ε, neglecting the second-order noise terms in z_q (which are very small), will be obtained as

$$\begin{aligned} \varepsilon &= z - z_q \\ &= x\sum_{i=1}^{H} (w_{2,i}v_{1,i1} + w_{1,i1}v_{2,i}) \\ &\quad + y\sum_{i=1}^{H} (w_{2,i}v_{1,i2} + w_{1,i2}v_{2,i}) \end{aligned} \tag{7}$$

$$z_q \simeq \begin{cases} 1 & \{(\boldsymbol{W}_2 + \boldsymbol{V}_2) \times \boldsymbol{I} \ge 1 \ \&\ (\boldsymbol{W}_1 + \boldsymbol{V}_1) \times \boldsymbol{P} \ge 1\}\ OR\ \{(\boldsymbol{W}_2 + \boldsymbol{V}_2) \times (\boldsymbol{W}_1 + \boldsymbol{V}_1) \times \boldsymbol{P} \ge 1\} \\ (\boldsymbol{W}_2 + \boldsymbol{V}_2) \times \boldsymbol{I} & \{(\boldsymbol{W}_2 + \boldsymbol{V}_2) \times \boldsymbol{I} \le 1\} \quad \& \quad \{(\boldsymbol{W}_1 + \boldsymbol{V}_1) \times \boldsymbol{P} \ge 1\} \\ (\boldsymbol{W}_2 + \boldsymbol{V}_2) \times (\boldsymbol{W}_1 + \boldsymbol{V}_1) \times \boldsymbol{P} & \{(\boldsymbol{W}_2 + \boldsymbol{V}_2) \le 1\} \quad \& \quad \{(\boldsymbol{W}_2 + \boldsymbol{V}_2) \times (\boldsymbol{W}_1 + \boldsymbol{V}_1) \times \boldsymbol{P} \le 1\} \end{cases} \tag{4}$$

doi: 10.1049/iet-ipr.2011.0079

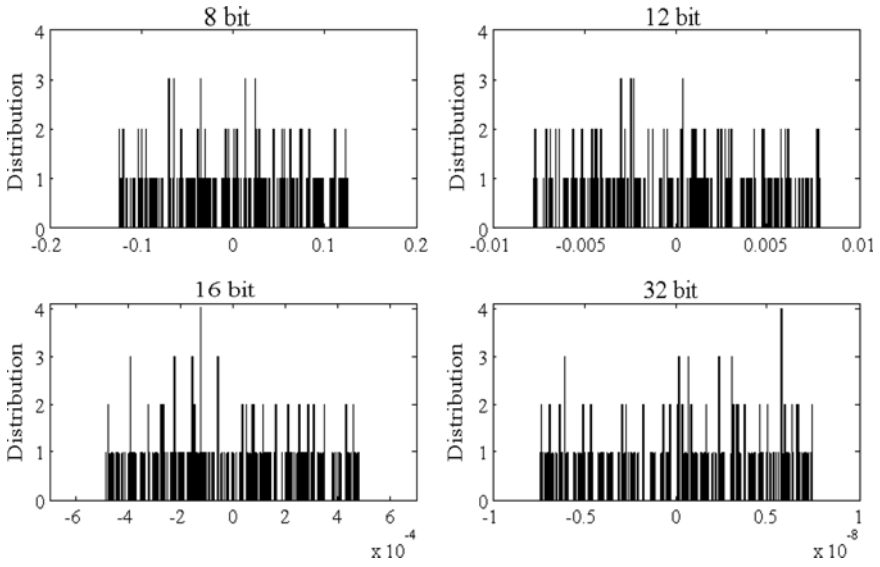

Fig. 4 *Round-off (quantisation) noise distribution for different number of bits*

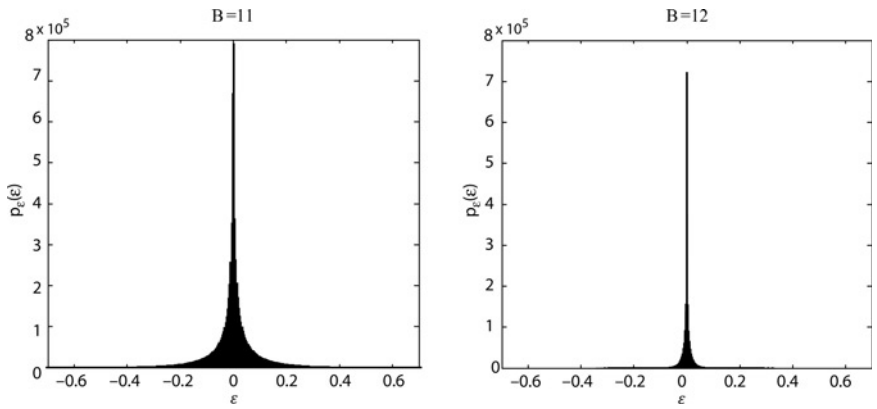

Fig. 5 *Distribution of output error ε owing to weight quantisation*

The distribution of the weights $w_{2,i}$ and $w_{1,ij}$ is shown in Fig. 3. In addition, the inputs applied to the network (pixels addresses) are normalised, hence x and y have uniform distribution in the interval [0, 1]. We observe from (7) that the round-off noise (which depends on the number of assigned bits B), number of hidden layer neurons (H) and the weights (W) have key roles in ε distribution. Fig. 5 demostrates the distribution of ε for $B = 11$ and $B = 12$ bits obtained by experiments.

The error ε, which is the output error owing to quantisation, is added to the network error (difference between the target value and network value). When the total error becomes more than 0.5, it will change the final result from '1' to '0' or vice versa. As observed from Fig. 5, $B = 11$ bits is more likely to cause this condition. For $B \geq 12$ bits, the quantisation error is not large enough to affect the final network decision. Therefore the number of bits assigned to the weights is chosen as $B = 12$. Further, our experiments show that 6 bits are enough to represent the mantis.

In Fig. 6, we have shown the PSNRs of some reconstructed sample images (shown in Fig. 7) against the number of bits assigned to the weights. The PSNR is defined as follows

$$\text{PSNR (dB)} = 10 \log_{10}\left(\frac{1}{\text{MSE}}\right) \tag{8}$$

where MSE is computed as

$$\text{MSE} = \frac{1}{\text{MN}} \sum_{i=1}^{MN} (z_i - t_i)^2 \tag{9}$$

In the above, z_i and t_i are the network actual output and the desired output (target; 0 or 1) of the ith input, respectively, and $M \times N$ is the size of image.

It is observed from Fig. 6 that the number of bits impacts on PSNR which is also related to image quality. However, this effect is not vital for 12 bits and more. By decreasing the number of representing bits from 16 to 12, the PSNR reduces slightly. As shown in Fig. 6, 12 bit representation is the breaking point of the curves. Thus, in consistence with the analysis and experiments, 12 bit representation is an optimum choice for our compression method.

2.3 Adaptive approach

The compression rate varies from one image to another, based on the number of weights that itself depends on the number of hidden layer neurons. The number of appropriate neurons that yields the best performance from the subjective and objective viewpoints can be obtained experimentally. It is obvious that the number of neurons for a complex image is more than that of a simple one.

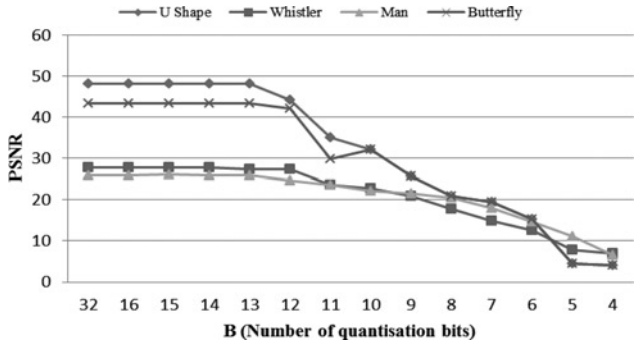

Fig. 6 *PSNRs of some reconstructed images against the number of bits assigned to NN weights*

In this study, the objective criterion is to achieve MSE < 0.006. It is evident that less MSE results in more PSNR and less classification error (CE) which is defined as the number of pixels wrongly detected to the total number of pixels. MSEs less than 0.006 achieve PSNRs more than 22 dB (8) and CEs less than 1%. The experiments show that PSNR > 22 dB results in high-quality image from the subjective viewpoint.

It will be useful to apply an adaptive approach to exploit the number of neurons automatically. For this purpose, we need to extract several features from image to apply to the adaptive algorithm so that the algorithm output will be the number of neurons. The features should measure the complexity of image which will be explained later.

In order to find the adaptive algorithm, we experimentally find the appropriate number of neurons for compression of 57 different images (18 images from [31] and 39 images from [32]) based on the criteria mentioned above. We also compute the features of each image. Then, the feature matrix and number of neurons are used for training the regression function. After training, the obtained regression function is applied to find the number of neurons of the other images (>4000) based on their calculated features. Here, we used non-linear regression model that shows better performance than the linear one. To train the regression function, we used an LM-based fitting neural-network as the non-linear regression method. Note that owing to the highly non-linear characteristics of the problem, our adaptive method does not necessarily achieve the optimum results. However, the results show good match between the number of neurons obtained from the experiments and the one computed from the regression.

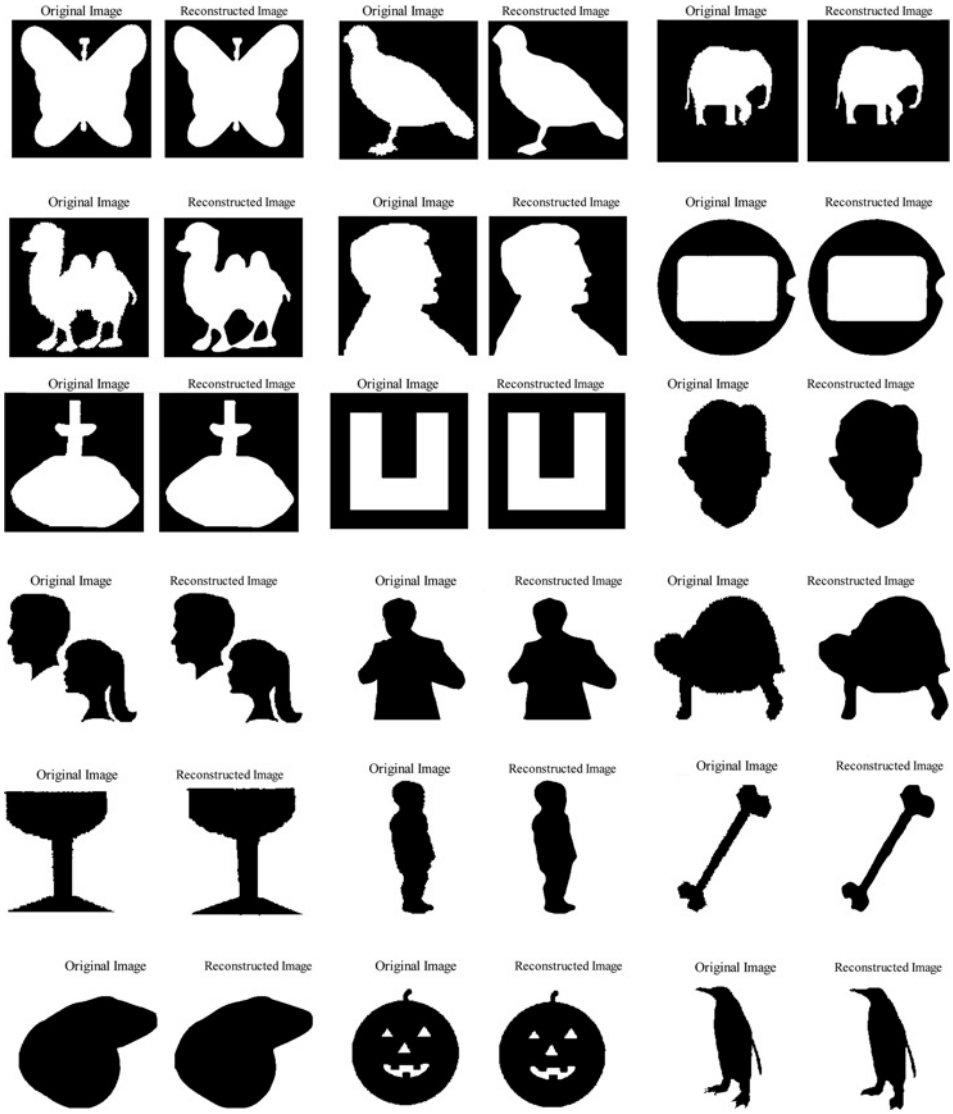

Fig. 7 *Experimental results of the proposed method evaluated on SIID [31]*

Images name from left to right and top to bottom: Butterfly, Bird, Elephant - Camel, Penny, Circle square - Fountain, U Shape, Face - Boy Girl, Man, Turtle - Glass, Child, Bone - Whistler, Jack o' Lantern, and Penguin

doi: 10.1049/iet-ipr.2011.0079

Image	H	PSNR (dB)	Image	H	PSNR (dB)	Image	H	PSNR (dB)
	33	24.8		24	21.16		38	22.72
	35	20.4		30	23.93		18	23.19
	27	26.64		10	∞		21	23.14
	31	21.42		15	25.15		24	20.9
	16	21.2		26	23.74		21	22.14
	12	25.49		41	22.7		28	24.1

Fig. 8 *Number of neurons in the hidden layer (H) of NN for different sample images from SIID [31] obtained experimentally and PSNRs of the reconstructed images*

Feature	Correlation	Description
Activity	0.796	defined in (10).
Hit or Miss Transform	0.731	The patterns used in the experiments are sensitive to 90° angles, such as: 
$\frac{\sum_{i=1}^{M}\sum_{j=1}^{N}(A_{ij}-C(A_{ij}))}{\sum_{i=1}^{M}\sum_{j=1}^{N}C(A_{ij})}$	0.560	This feature measures the protrusions; C(.) computes the convex hull of image.
$\frac{\sum_{i=1}^{M}\sum_{j=1}^{N}A_{ij}\otimes\{B\}^{\infty}}{\sum_{i=1}^{M}\sum_{j=1}^{N}A_{ij}}$	0.801	This feature reveals the structural complexity of the image; $\otimes$ indicates "thinning" operation and B is the structural element.
$\frac{\sum_{i=1}^{M}\sum_{j=1}^{N}((A_{ij}-S(A_{ij}))\ominus B)}{\sum_{i=1}^{M}\sum_{j=1}^{N}A_{ij}}$	0.793	This feature indicates the number of semi-objects inside an object; $S(.)$ computes the skeleton of the object and $\ominus$ is denotes "erosion". B is an appropriate structural element.
$Euler\left(A_{ij}-(A_{ij}\ominus B)\right)$	0.75	This feature reveals some information about the complexity of protrusions of the image; Euler is the indication of Euler number of a binary image, (it is the total number of objects in the image minus the total number of holes in those objects).
$\sum_{i=1}^{M}\sum_{j=1}^{N}Sobel\left(Hough(A_{ij})\right)$	0.74	This feature, similar to activity criterion, is used as another general complexity measure of the image. Hough filter is sensitive to lines and curves and this can be used for complexity measurement.

Fig. 9 *Some proposed complexity measures for binary images*

Fig. 8 demonstrates the number of hidden layer neurons (H) for different images obtained experimentally as well as the PSNRs of corresponding reconstructed images. We observe that the number of neurons is proportional to image complexity.

2.4 Feature selection

We propose 25 features to represent binary image complexity. Fig. 9 depicts several features.

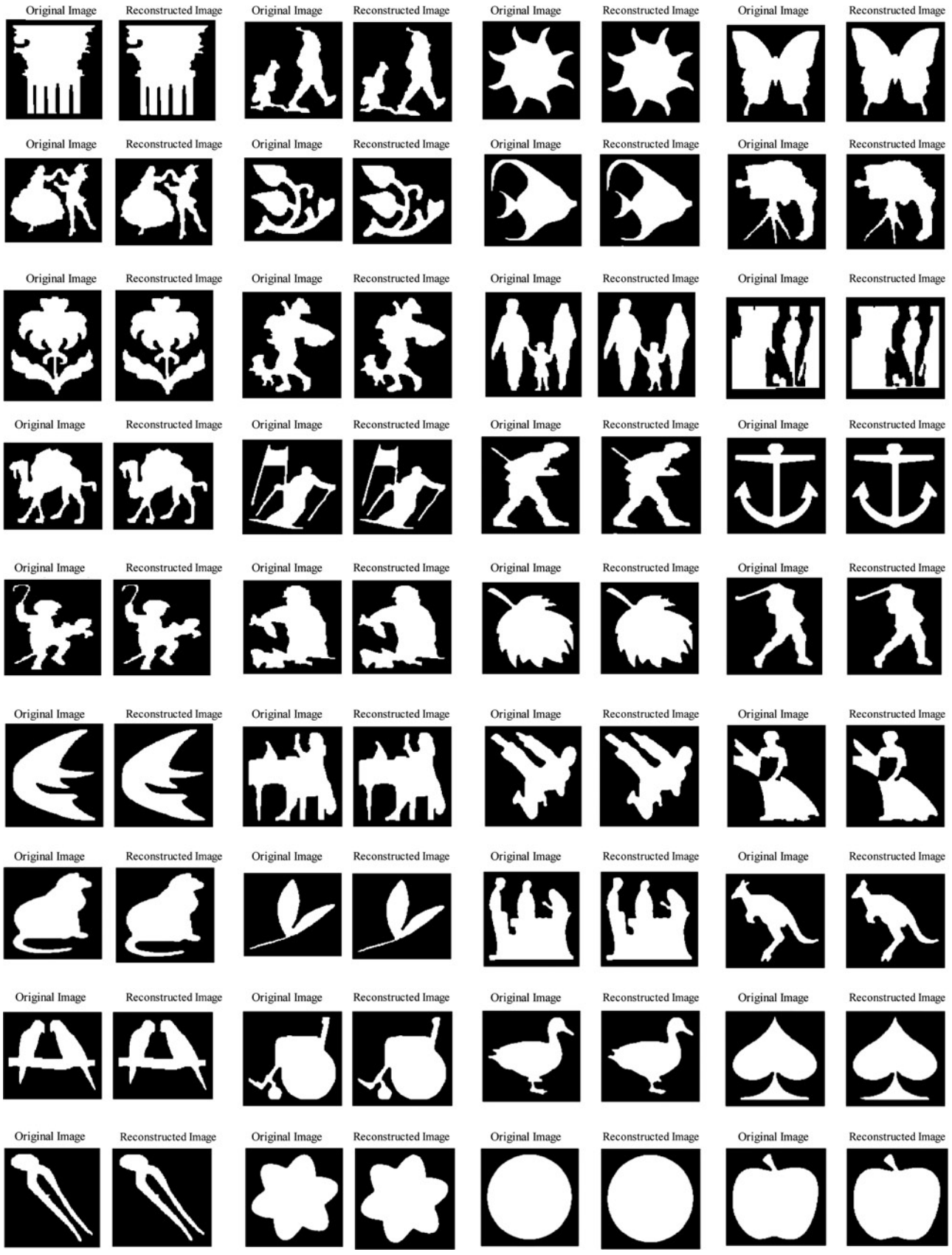

Fig. 10 *Experimental results of the proposed method evaluated on LBID [32]*

doi: 10.1049/iet-ipr.2011.0079

Activity [7] is the subjective assessment of an image given by

$$\text{Activity} = \sum_{x=1}^{M/2}\sum_{y=1}^{N/2}\left[\sum_{k=-1}^{1}\sum_{l=-1}^{1}(f(2x,2y)-f(2x-k,2y-l))^2\right] \tag{10}$$

where $f(x,y)$ specifies the intensity of the (x,y)th pixel. The above feature measures the difference between the centre pixel intensity with its neighbours in a 3×3 block. Images with high activity values seem more complex and their compression rates will be less than the images with low activity. The experimental results verify the above statement.

Pattern-based template matching is another complexity measure criterion. Our experiments reveal that images with straight borders, 90° corners and geometrical shapes are capable of compression with less number of hidden layer neurons. Thus, we use some specific patterns for this purpose. These patterns and their rotations and complements are matched with the image using a hit-or-miss transform. An image containing more number of a specific pattern can be represented with lower number of neurons. In our adaptive approach, we used three patterns that measure the 90° corners where one of them is shown in Fig. 9.

There are other complexity measures proposed in Fig. 9. These criteria show different aspects of binary image. For example, an image with a lot of small protrusions, narrow isthmuses, complex structure or different objects in a background are much more complicated than a generic geometrical shape such as square or circle. Hence, we suggested some criteria to measure these characteristics of image.

Further, to evaluate the effectiveness of these features, a correlation criterion has been used (which is shown in Fig. 9). This index shows the linearity of relation between the features and the number of neurons obtained experimentally. High correlation value means that the selected feature can better predict the number of neurons in the adaptive algorithm.

3 Experimental results

In this section, the performance of the proposed method is assessed. We used two databases. Eighteen images shown in Fig. 7 are selected from SIID (shape indexing of image databases) available in [31]. The second database is LBID (large binary image database) available in [32] which contains 4155 images. Thirty-six images selected from this database are also shown in Fig. 10 where most of them have complex shapes (to evaluate the capability of our algorithm for complex images).

Eighteen images of SIID (Fig. 7) and 39 images from LBID (not shown in the paper) were used to train the adaptive algorithm. The number of assigned bits for weights was set to 12 according to the discussion of the section 2.2.

In the experiments, we have used 18 images of SSID and all images of LBID (totally 4173 images) to evaluate the performance of the proposed compression method.

Fig. 7 demonstrates SIID images and their reconstructed counterparts using the proposed method. We observe that the decompressed images have high quality. The PSNRs are shown in Fig. 8. Further, the compression rates are provided in Table 1. From this Table, we observe that the new method gives compression ratios in the range of 43.8–171.8 which are high values.

Generally, the compression rate before Huffman coding is computed as

$$\text{CR} = \frac{M \times N}{L} \tag{11}$$

where L is total number of bits assigned to the weights computed as

$$L = (4H + 1) \times B \tag{12}$$

In the above, four denotes the number of weight sets; two sets for inputs to hidden layer, one for hidden layer bias, one for

Table 1 Compression ratios of several methods evaluated on SIID [31]

Image	Quad tree	TIFF					PDF	Proposed
		LZW	Pack Bit	Deflate	CCITT G3	CCITT G4	JBIG2	
butterfly	2.09	4.76	2.37	6.27	3.4	12.01	20.7	**50.0**
bird	1.92	5.72	3	7.24	3.52	11.67	21.7	**82.9**
elephant	3.48	6.25	3.39	8.66	3.68	13.65	25.9	**46.0**
camel	1.47	6.18	3.45	9.14	3.63	12.85	15.9	**50.0**
penny	4.72	6.06	3.12	8.03	3.67	13.43	26.4	**58.2**
circle-square	1.63	5.1	2.5	6.5	4.0	12.6	22.4	**96.5**
fountain	2.72	6.39	3.32	7.85	3.61	13.3	21.4	**64.6**
u shape	3.6	6.43	2.71	17.14	3.68	16.45	56.5	**171.8**
face	2.55	5.93	3.1	8.8	4.3	14.4	28.3	**82.9**
boy girl	2.37	4.8	2.4	7.0	3.8	12.6	22.2	**60.2**
man	2.13	5.9	3.2	8.3	4.4	14.08	29.2	**115.5**
turtle	2.98	5.5	2.7	7.9	4.1	13.2	22.1	**72.6**
glass	3.07	6.1	3.0	9.2	4.4	13.7	23.3	**108.4**
child	2.59	6.6	3.2	9.7	4.3	15.8	30.9	**67.1**
bone	2.16	6.6	3.3	8.8	4.7	15.3	28.2	**82.9**
whistler	3.01	6.4	3.7	7.8	4.6	14.4	29.4	**143.8**
jack-o'-lantern	2.89	5.5	2.9	7.2	4.2	14.1	24	**43.8**
penguin	2.25	5.9	3.2	8.3	4.4	14.8	25.7	**62.3**
average	2.65	5.9	3.0	8.5	4.0	13.8	26.3	**81.1**

hidden layer to output neuron; and one denotes the bias of output neuron.

We note that it is not necessary to send the number of neurons obtained from the adaptive algorithm, since it is simply computed from (12) at the decompression phase.

As an example of compression rate computation, we consider U shape image. The number of neurons in the hidden layer was set to 10 (Fig. 8). Hence, the total number of weights is 41 Assigning 12 bits for each weight results in totally 492 bits for weights representation. Consequently, without Huffman coding the compression rate is obtained as $256 \times 256/492 = 133.2$. However, after Huffman coding the compression ratio will increase to 171.8.

For comparison, the compression ratios of some common methods are also presented in Table 1. There are some standards for bi-level compression such as CCITT G3, CCITT G4, JBIG1 and JBIG2 [18–21]. It should be noted that these methods are lossless. However, we could not find any related and available lossy works in the field of binary image compression.

The JBIG2 standard discusses about the decoding part while it does not provide any comments on the image encoding. Therefore different implementations of JBIG2 can be developed. We have used all available tools.

The compression ratios of different TIFF compression schemes have been computed using GNU Gimp software and those of JBIG2 are obtained from JBIG2 encoder in AGL software. JBIG-KIT has been used as the JBIG1 encoder on Linux platform, however there was not much difference between JBIG1 and JBIG2 results which is also reported in [17]. JBIG2 as the state-of-the-art technique in this field, segments the input image into three different regions: text, halftone and generic area and uses different technologies for each region type. JBIG2 supports both lossy and lossless approaches and it is possible to use it in both cases.

We have also examined quad-tree method [33, 34]. The authors in [33] reported near to Lempel–Ziv–Welch (LZW) performance for their hybrid technique evaluated on grey scale images. However, the lower performance for binary images owing to much higher overhead of the quad-tree can be predicted. On the other hand, having 1 bit/pixel of information for binary images makes quad-tree technique less successful in this case.

In the experiments presented in Fig. 7 and Table 1, the size of images was 256×256 pixels. We have also tried images with different sizes. The results show that if the reconstructed image quality does not change, then the compression rate increases linearly by increasing the image size. Note that, as the image size grows, the nominator of the compression rate in (11) increases with power of 2, whereas the denominator increases linearly with the size because of increase in the implemented neural-network complexity (number of neurons). The result will be the linear increase of compression ratio with respect to image size.

Furthermore, we examined the proposed method on 4155 images of LBID [32] which have different shapes from simple to complex. Several (36) of them and their corresponding reconstructed counterparts are shown in Fig. 10. The images shown in this figure are of size 128×128 pixels. Moreover, Fig. 11 depicts the PSNR, classification rate, compression ratio and number of hidden layer neurons computed from adaptive algorithm for these images. We observe that for complex images, the number of neurons is high which leads to less compression ratio and vice versa. Also, in Table 2, we have compared the

image	PSNR (dB)	CE (%)	H	CR	image	PSNR (dB)	CE (%)	H	CR	image	PSNR (dB)	CE (%)	H	CR
	25.7	0.3	48	10.3		23.6	0.4	48	11.4		33.1	0	13	27.8
	22.8	0.5	27	21.9		27.5	0.2	29	19.2		41.6	0	14	30.4
	22.0	0.6	14	25.1		32.3	0.1	48	7.4		22.6	0.6	26	22.9
	25	0.3	31	18.5		24.3	0.4	17	25.1		33	0	23	23.6
	22.0	0.6	50	10.7		25.1	0.3	50	8.1		37.9	0	27	13.6
	32.8	0.1	46	7.5		27.4	0.2	28	20.2		32	0.1	25	22.6
	42.3	0	18	28.4		30.0	0.1	47	11.4		33.5	0	29	16.1
	21.9	0.7	48	8.8		40.7	0	21	16.7		30.4	0.1	15	32.8
	19.7	1.1	43	8.6		33.0	0.1	27	16.6		39.1	0	6	79.8
	21.6	0.7	44	12.8		25	0.3	27	21.8		∞	0	21	26.1
	24.8	0.3	40	14.8		28.1	0.2	16	35.1		∞	0	11	48.6
	24.1	04	41	9.3		28.0	0.2	37	14.8		32	0.1	15	34.0

Fig. 11 *Experimental results of the proposed method evaluated on LBID [32] CE, H and CR denote classification error, number of hidden layer neurons and compression ratio respectively*

compression ratios of several methods evaluated on the mentioned LBID images. It is observed that the proposed method achieves higher compression ratios than the other methods. Note that the higher average CR of Table 1 (SIID images) compared to that of Table 2 (LBID images) is because of two reasons: higher size of SIID images (256×256) than that of Table 2 (128×128) and simpler shapes of SIID compared to the LBID images of Table 2.

In Fig. 12, the results of experiments on the two databases (totally 4173 images) are classified based on the PSNRs and our observations. It should be noted that the subjective classification of reconstructed images is related to the subjective evaluation of quality and the PSNR of image. Based on the results obtained from the experiments on 4173 images and our observations, we divided the reconstructed images into five categories according to the PSNR value and subjective quality as follows: As can be seen from the table, 8.8% of decompressed images are 'lossless', 63% of them are considered as 'near-to-lossless' and 22.3% as 'good', which constitute 94.1% of total images. We considered the images with no reconstruction error as 'lossless' where the PSNR is infinite. Those images that are visually lossless and the error in the reconstructed image cannot be detected easily are labelled as 'near-to-lossless'. These images have PSNR values more than 27 dB, where the classification error for a 256×256 image is less than 0.2% hard to be detected by human eye. 'Good' quality refers to those reconstructed images that have PSNRs in the range 22–27 dB. These images show very good quality since the errors cannot be easily seen or can be neglected in the reconstructed image. Moreover, 'fair' quality refers to the images that some errors can be detected by human eye but the quality of reconstructed image is still acceptable. Our investigations show that the PSNRs between 18 and 22 dB show 'fair' quality. The images with some missed details, which cannot be neglected, are considered as 'poor'. The PSNRs of such images are in the interval [14–18] dB. Those images that have PSNRs less than 14 dB are considered as 'defaced'. These images usually do not convey any relevant information about the original images.

doi: 10.1049/iet-ipr.2011.0079

Table 2 Compression ratios of several methods evaluated on LBID [32]

Image	TIFF					PDF	Proposed
	LZW	Pack Bit	Deflate	CCITT G3	CCITT G4	JBIG2	
202122	2.5	1.3	3.6	1.7	4.4	9.8	**10.3**
dec069ee	2.5	1.3	3.2	2.1	5	10.6	**21.9**
15b_starburst	2.7	1.5	3.2	2.3	5.2	10.4	**25.1**
anmo8	2.5	1.3	3.3	2.1	5.3	10.9	**18.5**
dec065h	2.2	1.2	2.8	2	4.4	**10.8**	10.7
ofl007e	1.8	1.1	2	1.6	3	**11.2**	7.5
anm01	2.4	1.3	2.9	2	4.7	10.7	**28.4**
msl050c	1.9	1.1	2.5	1.7	3.8	**9.7**	8.8
ofl028k	2	1.1	2.3	1.7	3.7	**10.6**	8.6
msl080c	2.4	1.4	3	2.1	4.6	7.7	**12.8**
family	2.2	1.1	3.1	1.9	4.9	10.1	**14.8**
ots026h	1.9	1.1	2.6	1.7	3.6	**11.9**	9.3
sit078n	2.3	1.3	2.9	2	4.5	8.3	**11.4**
sposk079	2.2	1.2	2.7	2	4.4	8.0	**19.2**
msl046b	1.8	0.9	2.2	1.5	3.3	**10.2**	7.4
anchor	2.3	1.2	3	1.9	4.3	10.0	**25.1**
smr021d	2	1.1	2.4	1.6	3.6	**10.2**	8.1
201872	2.6	1.4	3.3	2.2	4.9	10.2	**20.2**
leaf	2.7	1.6	3.2	2.2	4.8	9.9	**11.4**
osp009e	1.7	0.9	2.2	1.4	3.2	10.0	**16.7**
moon	2.3	1.3	2.7	1.9	4.1	9.5	**16.6**
1245play	2.4	1.2	3.6	2	5.1	7.4	**21.8**
karate_1	2.5	1.4	3	2.2	4.9	7.9	**35.1**
g0180443	2.5	1.4	3.1	2.2	4.9	9.5	**14.8**
cat_1	2	1.1	2.5	1.7	3.7	9.4	**27.8**
ofl006g	2.6	1.5	2.9	2.2	4.2	5.4	**30.4**
ots022p	2.5	1.3	3.5	2	5.3	7.4	**22.9**
ai01042a	2.9	1.7	3.3	2.4	5.3	6.3	**23.6**
anm03	2.2	1.3	2.6	2	3.8	10.3	**13.6**
202137	2.7	1.6	3.6	2.3	5.4	7.0	**22.6**
duck_2	2.8	1.7	3.2	2.3	4.5	8.2	**16.1**
spades	2.4	1.4	2.8	2	4.5	7.6	**32.8**
tool2	2.4	1.4	2.8	2	4.5	9.8	**79.8**
29b_starburst	2.8	1.6	3.4	2.2	5.2	9.5	**26.1**
circle	2	1.2	2.4	1.7	3.4	6.8	**48.6**
apple	2.7	1.6	3.4	2.1	5.2	11.6	**34**
average	2.3	1.3	2.9	2.0	4.4	9.3	**21.2**

Names correspond to the images of Fig. 11 from top to bottom and left to right order

The last two categories constitute near 1% of total images. These images are usually the result of early stopping of the training algorithm (0.65%) (which is inevitable for large-scale experiments) or under estimation of number of neurons in the adaptive algorithm (0.55%). Early stopping is usually owing to mal-initialisation of the network that causes zero gradient coefficients in the first iterations.

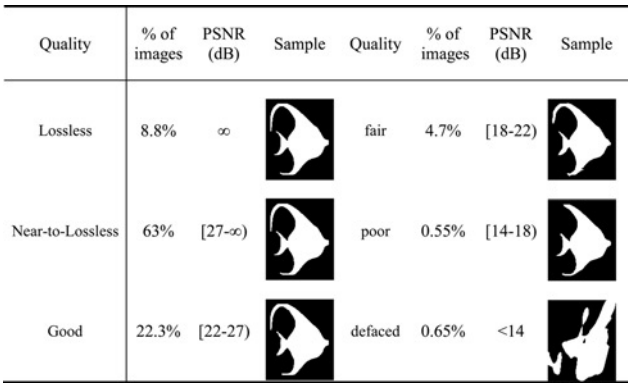

Fig. 12 *Subjevtive quality of experiment results on whole LBID (4155 images)*

3.1 Computational complexity

We used Intel® Core™2Due processor E8400 (3.0 GHz) with 2 GB of memory on a Windows XP operating system. We also utilised MATLAB 2009 software for simulations. Neural-network toolbox was also applied for neural-network part of our approach.

Training time for different images varied from 10 to 30 seconds depending on the images structures. In the decompression phase, the reconstruction time was about 15–40 ms.

Utilisation of a fast and reliable training algorithm is crucial in our approach. We used LM algorithm for training the network which constitutes the most time consuming part of our technique. LM belongs to those back propagation algorithms which deploy the second derivative of error in

the network with respect to the weights and biases, that is, the calculation of Hessian matrix is needed. However, in LM algorithm, the Hessian matrix is estimated, without direct calculation. LM uses Jacobian matrix for this purpose which has been previously computed during the calculation of the first derivative of the network weights. Therefore LM algorithm is considered as one of the best back propagation algorithms from the computational complexity view point.

In summary, the characteristics of LM algorithm which makes it appropriate for our problem can be listed as follows:

1. LM yields the best MSE performance, which is related to PSNR.
2. It fits very well in small to medium problems with usually less than 50 neurons and one hidden layer like as our case.
3. It is usually considered as the fastest training algorithm.

4 Conclusions

In this paper, binary image compression using a new scheme of neural-network was investigated. In our method, the weights of trained neural-network are used instead of image intensities. High compression ratios as well as high PSNRs were obtained using the proposed method. We have also analysed the effect of quantisation noise in neural-network deployment of image compression. Further, we used activity, pattern based criteria and some complexity measures to adaptively obtain high compression rate. It was mentioned that the compression ratio increases linearly with the image size. The proposed structure of neural-network utilisation has the capability to compress binary image in efficient way.

5 References

1 Gonzalez, R.C., Woods, R.E.: 'Digital image processing' (Pearson Education, 2008, 3rd edn.)
2 Shannon, C.: 'Coding theorems for a discrete source with a fidelity criterion', *IRE Natl Conv Rec.*, 1959, **7**, pp. 325–350
3 Jayant, N., Johnston, J., Safranek, R.: 'Signal compression based on models of human perception', *Proc. IEEE*, 1993, **81**, (10), pp. 1385–1421
4 Zheng, Y.: 'An improved fractal image compression approach by using iterated function system and genetic algorithm', *Comput. Math. Appl.*, 2006, **51**, pp. 1727–1740
5 Fisher, Y.: 'Fractal image compression' (Springer-Verlag, New York, 1995)
6 Gersho, A., Gray, R.: 'Vector quantization and signal compression' (Springer, 1991, 1st edn.)
7 Veisi, H., Jamzad, M.: 'A complexity-based approach in image compression using neural networks', *Signal Process.*, 2009, **5**, pp. 82–92
8 Salomon, D.: 'A concise introduction to data compression' (Springer, 2008, 1st edn.)
9 Wu, M.S., Lin, Y.L.: 'Genetic algorithm with a hybrid select mechanism for fractal image compression', *Digit. Signal Process.*, 2009, **20**, pp. 1150–1161
10 Xing-yuan, W., Fan-ping, L., Shu-guo, W.: 'Fractal image compression based on spatial correlation and hybrid genetic algorithm', *J. Vis. Commun. Image Represent.*, 2009, **20**, pp. 505–510
11 Franti, P.: 'Genetic algorithm with deterministic crossover for vector quantization', *Pattern Recognit. Lett.*, 2000, **21**, pp. 1568–1575
12 Dasgupta, D., Hernandez, G.: 'An evolutionary algorithm for fractal coding of binary images', *IEEE Trans. Evol. Comput.*, 2000, **4**, (2), pp. 172–181
13 Doney, R.D., Haykin, S.: 'Neural network approaches to image compression', *Proc. IEEE*, 1995, **83**, (2), pp. 288–302
14 Pandian, S.I.A., Anitha, J.: 'A neural network approach for color image compression in transform domain', *Int. J. Recent Trends Eng.*, 2009, **2**, (2), pp. 152–154
15 Soliman, H.S., Omari, M.: 'A neural networks approach to image data compression', *Appl. Soft Comput.*, 2004, **6**, pp. 258–271
16 Grailu, H., Lotfzad, M., Sadoghi-Yazdi, H.: '1-D chaincode pattern matching for compression of bBi-level printed farsi and arabic textual images', *Image Vis. Comput.*, 2009, **27**, pp. 1615–1625
17 Alcaraz-Corona, S., Rodriguez-Dagnino, R.M.: 'Bi-level image compression estimating the markov order of dependencies', *IEEE J. Sel. Top. Signal Process.*, 2010, **4**, (3), pp. 605–611
18 JBIG2 Final Draft International Standard: ISO/IEC JTC1/SC29/WG1N1545, December 1999
19 Information Technology – Coding of Audio-Visual Objects – Part 2: Visual, ISO/IEC Int. Std. 14496-2, 1999
20 Howard, P.G., Kossentini, F., Martins, B., Forchhammer, S., Rucklidge, W.J., Ono, F.: 'The emerging JBIG2 standard', *IEEE Trans. Circuits Syst. Video Technol.*, 1998, **8**, (7), p. 833-348
21 Coded representation of picture and audio information – lossy/lossless coding of bi-level images (JBIG2), ISO/IEC Int. Std. 14492, 2000
22 Wu, B.F., Chiu, C.C., Chen, Y.L.: 'Algorithm for compression compound document images with large text/background overlap', *IEE Proc.*, 2004, **151**, (6), pp. 453–460
23 Sanchez-Cruz, H., Bribiesca, E., Rodriguez-Dagnino, R.M.: 'Efficiency of chain codes to represent binary objects', *Pattern Recognit.*, 2007, **40**, (6), pp. 1660–1674
24 Liu, Y.K., Wei, W., Wang, P.J., Zalik, B.: 'Compressesd vertex chain codes', *Pattern Recognit.*, 2007, **40**, pp. 2908–2913
25 Aghtio, S.M., Forchhammer, S.: 'Context-based coding of bilevel images enhanced by digital straight line analysis', *IEEE Trans. Image Process.*, 2006, **15**, (8)
26 Xia, S., Boncel, C.G.: 'On the use of context-weighting in lossless bilevel image compression', *IEEE Trans. Image Process.*, 2006, **15**, (11), pp. 3233–3260
27 Makarov, A., Moniri, M.: 'Binary shape coding using finite automata', *IEE Proc.*, 2006, **153**, (5), pp. 695–702
28 Falkowski, B.J.: 'Lossless binary image compression using logic functions and spectra', *Comput. Electr. Eng.*, 2004, **30**, pp. 17–43
29 Huffman, D.A.: 'A method for the construction of minimum redundancy code', *Proc. IRE*, 1952, **40**, (10), pp. 1098–1101
30 Oppenheim, A.V., Schafer, R.W., Buck, J.R.: 'Discrete time signal processing' (Printice-Hall, 2010, 3rd edn.)
31 http://www.lems.brown.edu/vision/researchAreas/SIID/
32 http://www.lems.brown.edu/~dmc
33 Zhenliang, S., Frater, M.R., Arnold, J.F.: 'Quad-tree block-based binary shape coding', *IEEE Trans. Circuit Syst. Video Technol.*, 2008, **18**, (6), pp. 845–850
34 Khan, M., Ohno, Y.: 'A hybrid image compression technique using quad tree decomposition and parametric line fitting for synthetic images', *Adv. Comput. Sci. Eng.*, 2007, **1**, (3), pp. 263–283

doi: 10.1049/iet-ipr.2011.0079