Learning Random Networks for Compression of Still and Moving Images

Erol Gelenbe, Mert Sungur*
Christopher Cramer
Department of Electrical Engineering
Duke University
Durham, NC 27708-0291
E-mail: erol@ee.duke.edu
Tel: (919) 660–5442, Fax: (919) 660–5293

Summary

Image compression for both still and moving images is an extremely important area of investigation, with numerous applications to videoconferencing, interactive education, home entertainment, and potential applications to earth observation, medical imaging, digital libraries, and many other areas.

In this paper we describe our work on a neural network methodology to compress/decompress still and moving images. We use the "point-process" type neural network model we have developed [12, 13, 16] which is closer to biophysical reality than standard models, and yet is mathematically much more tractable. We currently achieve compression ratios of the order of 120 : 1 for moving grey-level images, based on a combination of motion detection and compression. The observed Signal-to-Noise-Ratio varies from values above 25 to more than 35. Our method is computationally fast so that compression and decompression can be carried out in real-time. It uses the adaptive capabilities of a set of neural networks so as to select varying compression ratios in real-time as a function of quality achieved. It also uses a motion detector which will avoid retransmitting portions of the image which have varied little from the previous frame.

Further improvements can be achieved by using on-line learning during compression, and by appropriate compensation of non-linearities in the compression/decompression scheme. We expect to go well beyond the 250 : 1 compression level for color images with good quality levels.

*Mr Sungur's work was supported by a NATO Science Fellowship at Duke University administrated by The Scientific and Technical Research Council of Turkey (TUBITAK), on leave from Department of Electrical and Electronics Engineering, Middle East Technical University, 06531 Ankara, Turkey

1 Introduction

As the volume of imaging data increases exponentially in a very wide variety of applications – including remote sensing, earth observation, medical imaging, digital libraries and documents, HDTV, entertainment and film, and videoconferencing – and as the needs for storing, retrieving and transmitting images expand, digital image compression is becoming an even more crucial technology. Many of these application areas – including earth observation, videoconferencing and many military applications – deal with sequences of images which represent some form of motion. For instance, sequences of pictures taken by a satellite each time it passes over nearly the same stretch of territory, after appropriate repositioning and compensation, are successive instances of the same scene containing changes due to the motion of objects (vehicles, for instance), or due to changing meteorological conditions. Thus compression can take great advantage of the fact that image sequences need only keep track of *changes* which occur from one frame to the next.

In some areas (such as medical imaging) it is more customary to deal with grey-level images. In other areas of application, one deals overwhelmingly with colour images (as in entertainment). The quality of a processed or compressed image is judged quite differently, whether one deals with grey-level or with colour. In the case of color, acceptable image quality will largely depend on the application. For instance, in HDTV one would be unhappy with a change in skin pigmentation (a greenish face does not look too good ...), while the change in a dress' colour may not matter too much.

Lossless compression is adequate when low compression ratios are acceptable. Very substantial compression ratios can only be achieved with *lossy* compression schemes. Many applications will accept lossy compression, as long as the resulting quality is good. In some critical applications – such as medical imaging and military observation – loss may not be tolerated. However even in those applications, compressed versions of archival images may be conveniently used for remote interrogation and fast access. The aim is of image compression is to encode images or image sequences into as few bits as possible with a decoding mechanism which reconstructs the original image with an acceptable visual and/or informational quality. Another issue in image compression and decompression is its speed, especially in real-time applications, or in those in which the rate at which the source produces data is very high. It is therefore often important to be able to carry out compression and decompression "on the fly" without additional delay in conveying the image.

In this paper we will describe a method for compressing and decompressing still and moving images. For moving image sequences of grey-level images, we obtain better than 110 : 1 compression levels with 20 *to* 30 Signal to Noise Ratio (SNR). We use a learning algorithm for the "random neural network" model (Gelenbe 1989, 1990, 1993 [12, 13, 16] [1]) to "teach" a set of networks to compress at different compression levels. A schematic representation of the complete method we propose is shown in Figure 1. The method uses a simple motion detection scheme, together with the set of learning neural networks for compression and decompression.

In the sequel we first describe the problem, then review the literature, after which we describe our method together with measurements describing the resulting compression levels, the SNR of reconstructed images. We also provide an indication of the data transmission rates for the schemes we develop. This last metric is particularly relevant when images are transferred over networks, since the nature of the traffic determines the performance levels which can be expected and the appropriate traffic controls which may have to be imposed.

[1] This model has also been successfully applied to other applications including optimization [15] and image texture analysis and reconstruction [3, 4].

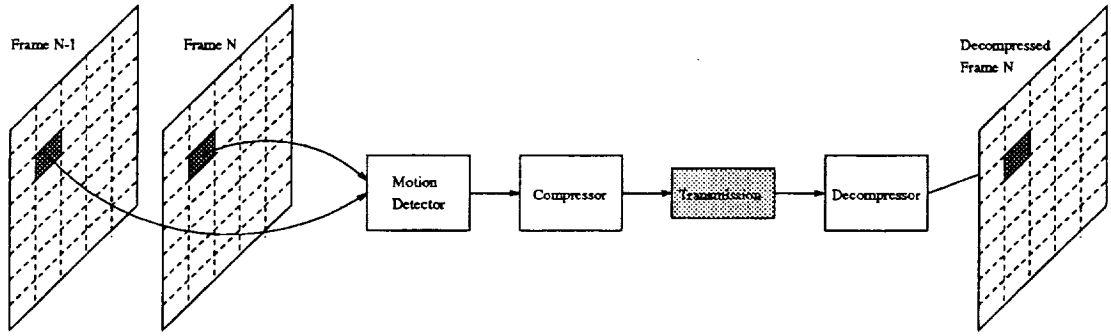


Figure 1: Block diagram of the complete compression scheme.

1.1 The Image Compression Problem

A *digital image* I is described by a function $f : Z \times Z \rightarrow \{0, 1, \ldots, 2^k - 1\}$ where Z is the set of natural numbers, and k is the maximum number of bits to be used to represent the gray level of each pixel. In other words, f is a mapping from discrete spatial coordinates (x, y) to gray level values. Thus, $M \times N \times k$ bits are required to store an $M \times N$ digital image. The aim of digital image compression is to develop a scheme to encode the original image I into *the fewest* number of bits such that the image I' reconstructed from this reduced representation through the decoding process is as similar to the original image as possible: *i.e.* the problem is to design a COMPRESS and a DECOMPRESS block so that $I \sim I'$ and $|I_c| << |I|$ where $|.|$ denotes the size in bits (Figure 2).

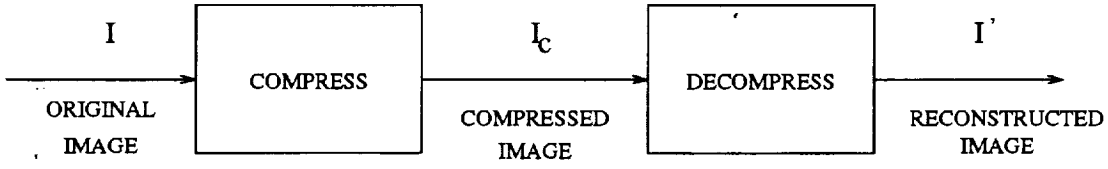


Figure 2: Image Compression Block Diagram

The similarity measure can vary for each application. Some applications may require the reconstructed image to be exactly the same as the original image, in which case the process is called *lossless compression*. In *lossy compression*, the peak signal-to-noise ratio or SNR is used as the measure of similarity or of dissimilarity, although it does not necessarily reflect visual quality. Assuming that the original and reconstructed images are represented by functions $f(x, y)$ and $g(x, y)$ of the pixel plane position (x, y), respectively, the SNR is defined by:

$$\text{SNR} = 10 \log_{10} \frac{(2^k - 1)^2}{e^2_{\text{rms}}} \tag{1}$$

where the root-means-square error is

$$e^2_{\text{rms}} = \overline{e^2} = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\, g(x, y) - f(x, y) \,]^2 \tag{2}$$

When moving images are concerned, the compression ratio may vary dynamically with the specific image or image portion being transmitted, since some advantage will be taken of the existence or non-existence of significant motion in successive image frames. However the SNR metric will still be relevant to the evaluation of the resulting quality.

1.2 State-of-the-Art in Still and Moving Image compression

Image compression research generally addresses the basic trade-off between the reconstruction quality of the compressed image, the compression ratio, and the complexity and speed of the compression algorithm. The two currently accepted standards for still and moving image compression are JPEG ([34]) and MPEG ([25]). These schemes provide high compression ratios with good picture reconstruction qualities. However, the amount of computation required for both is generally too high for real-time applications. MPEG uses the following techniques: 1) RGB color space coding to YCrCb coding, this gives and automatic 2:1 compression ratio, 2) JPEG encoding based on discrete cosine transform and quantization followed by some lossless compression, which yields compression ratios as high as 30:1 with good image quality, and 3) Motion Compensation, in which a frame can be encoded in terms of the previous and next frames. However, these techniques severely limit the speed at which a sequence of images can be compressed.

Two classical techniques for still image compression are transform and sub-band encoding. In transform coding techniques the image is subdivided into small block each of which undergoes some reversible linear transformation (Fourier, Hadamard, Karhunen-Loeve, etc.) followed by quantization and coding based on reducing redundant information in the transformed domain. In subband coding ([35]), an image is filtered to create a set of images, each of which contains a limited range of spatial frequencies. These so-called subbands are then downsampled, quantized and coded. These techniques require much computation. Another common image compression method is vector quantization ([18]) which can achieve high compression ratios. A vector quantizer is a system for mapping a stream of analog or very high rate or volume discrete data into a sequence of low volume and rate data suitable for storage in mass memory, and communication over a digital channel. This technique mainly suffers from edge degradation and high computational complexity. Although some more sophisticated vector quantization schemes have been proposed to reduce edge effects ([30]), the computation overhead still exists. Recently, novel approaches have been introduced based on pyramidal structures [1], wavelet transforms [36], and fractal transforms [20]. These and some other new techniques [24] inspired by the representation of visual information in the brain, can achieve high compression ratios with good visual quality but are nevertheless computationally intensive.

The speed of compression/decompression is a major issue in applications such as videoconferencing, HDTV applications, videophones, which are all likely to be a part of daily life in the near future. Artificial neural networks [31] are being widely used as alternative computational tools in many applications. This popularity is mainly due to the inherently parallel structure of these networks and to their learning capabilities which can be effectively used for image compression.

Several researchers have used the Learning Vector Quantization (LVQ) network [23] for developing codebooks whose distribution of codewords approximates the probabilistic distribution of data which is to be presented. A Hopfield network for vector quantization which achieves compression of less than 4:1 is reported in [27]. A Kohonen net method for codebook compression is demonstrated in [29]; it seems to perform slightly better than another standard method of generating codebooks. Cottrell et al. ([8]) train a two-layer perceptron with a small of number of hidden units to encode and decode images, but do not report encouraging results about the performance of the network on previously unseen images. Using neural encoder/decoders has been suggested by many researchers such as [6]. In [10], the authors present a neural network method for finding coefficients of a 2-D Gabor transform. This 2-way function can then be quantized and encoded to give good images at compression of under 1 bit/pixel, and as low as 0.38 bits/pixel with good image quality in a particular case.

A feed-forward neural network model to achieve 16 : 1 compression of untrained images with $SNR = 26.9dB$ is presented in [26] by using four different networks to encode different "types" of images. A backpropagation network to compress data at the hidden layer and an implementation on a 512 processor $NCUBE$ are discussed in [32]. In [19], the authors perform a comparison of backpropagation networks with recirculation networks and the DCT (discrete cosine transform). The best results reported here are obtained with the DCT, then with recirculation networks and finally with backpropagation networks. An interesting feature

of this paper is that they show the basis images for the neural networks, which allows one to compare the underlying matrix transformations of the neural networks to that of the DCT. In [11], the authors present a VLSI implementation of a neuro vector quantization/codebook algorithm. In [28], the authors use a back-propagation based nested training algorithm to do compression. For images on which the network has already been trained (which is not specifically of practical use) the compression ratios and resulting qualities are as follows: 8:1 (SNR = 22.89dB), 64:1 (SNR=15.15dB) to 256:1 (SNR=10.44dB). For previously "unseen" images, results are given with the following ratios and qualities: 8:1 (SNR=18.13dB) to 64:1 (SNR=12.93dB). Our own results for "unseen" images provide substantially better quality, especially at the lower compression ratios (8:1 and 16:1). In [22], the authors suggest the use of a non-linear mapping function whose parameters are learned in order to achieve better image compression in a standard backpropagation network.

Motion detection and compensation are key issues when one deals with moving images. Motion compensation provides for a great deal of the compression in the MPEG standard. By using motion compensation, MPEG can code the blocks in a frame in terms of motion vectors for the blocks in the previous and/or next frames. To perform motion must be estimated using block matching over the area local to the block under consideration. Exhaustive searches which consider all possible motion vectors yield good results. However for large ranges, the cost of such a search becomes prohibitive and heuristic searches must be used. This also raises the problem that full motion compensation cannot be performed in real time since it requires the future frame to be known in advance. Partial motion compensation, in which blocks may be encoded only in terms of blocks in the previous frame, may be used. One should also note that the MPEG standard does not specify the method of motion compensation to be used and a neural solution to motion compensation problem in two dimensions has been examined. In [9], a neural network for motion detection is presented; however it only works for a one dimensional case and the authors state that problems arise when the approach is extended to two dimensional detection of edge motion. It appears this approach would involve a great deal of research before it could be usefully applied in moving picture compression. In [7], a neural network method for motion estimation is presented. Drawbacks include the assumption that displacement is uniform in the area of interest. This would be a problem in trying to estimate the motion of a human being in which motion vectors differ over subsets of the picture.

2 Still Image Compression with the Random Neural Network

One of the common neural approaches in image compression is to train a network to encode and decode the input data [8], so that the resulting difference between input and output images is minimized. The network consists of an input layer and an output layer of equal sizes, with an intermediate layer of smaller size in between. The ratio of the size of the input layer to the size of the intermediate layer is – of course – the compression ratio. More generally, there can also be several intermediate layers. The network is usually trained on one or more images so that it develops an internal representation corresponding not to the image itself, but rather to the relevant features of a class of images.

In our approach, both the input, intermediate and output image is subdivided into equal-sized blocks and compression is carried block by block (see Figure 3). This has the desirable effect of reducing the network learning time. It also achieves good generalization, since the blocks comprising a single test image are used as the training set. The amount of information representing the compression and decompression algorithm (i.e. the "weights") is also substantially reduced in this manner. We use a feedforward encoder/decoder random neural network with one intermediate layer as shown in Figure 8. The weights between the input layer and the intermediate layer correspond to the encoding or *compression* process, while the weights from the intermediate to the output layer correspond to the decoding or *decompression* process.

Our current results use 8×8 boxes, where each element is a byte. We encode the 8-bit gray level values as real numbers between 0 and 1, *i.e.* we map the $[0, 255]$ interval into the $[0, 1]$ interval since the grey level of each image pixel is transformed into a real-valued excitation level of a neuron (and vice-versa). The network

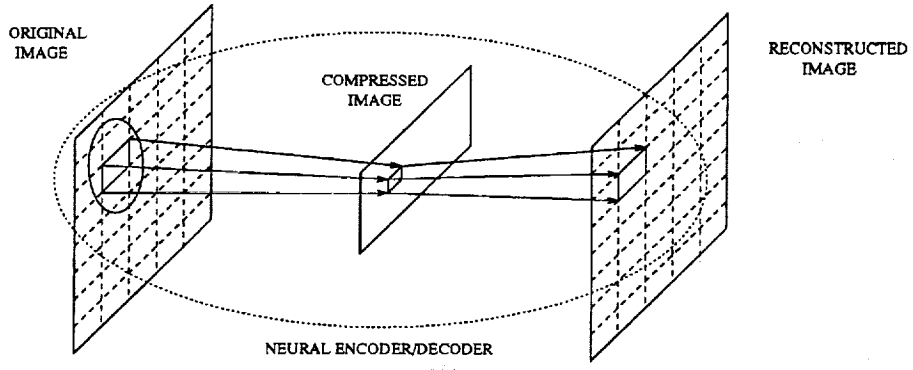


Figure 3: Compression of an arbitrarily large image using a neural encoder/decoder

is trained so as to minimize the squared error between the output and input values, thus maximizing the SNR, with the proviso that the image SNR is measured for quantized values in $[0, 255]$ while the neural network learning uses the corresponding real-valued network parameters. In all the results we report, both in this section and when we deal with moving images, our networks are trained using the algorithm described in [16] using a single image: the well-known 512×512 8-bit *Lena*. Indeed, we have found that *Lena* provides some of the best results for training the network. The network is then tested for a variety of images, and we have observed a reconstruction quality ranging from $SNR = 23dB$ to more than $30dB$ for $16:1$ compression (*i.e.* 0.5 bits/pixel). As an example, Figure 4 shows our results with $16:1$ compression for the 512×512 8-bit *Peppers* image [17].

PEPPERS original SNR = 27.82

Figure 4: Test results for $16:1$ compression (0.5 bit/pixel) with random neural network

2.1 Motion Detection

In many applications such as *videoconferencing*, sequences of image frames representing a moving scene are transmitted. Often, a substantial part of an image, such as the background, basically does not move – except for noise which may originate at various levels, including the imaging devices. On the other hand, the objects in the image move relative to the background, but this displacement be quite small between any two successive frames. We use these facts in order to perform motion detection. Specifically we examine the 8×8 boxes from successive frames F_{i-1}, F_i. Motion is sensed if the average grayscale value of a box in F_i differs from that of the corresponding box in frame F_{i-1} by more than a certain amount d. We have observed experimentally that the difference in the average grayscale value of a block that is perceptable to the human

eye is around around $d = 1$. Note that the box structure used throughout our compression scheme makes this approach possible as long as the box size is small enough. Indeed, a large box size would either make it highly improbable that motion has not occurred within any given box, or would render the detection process insensitive if accompanied by a large value of d.

We use the first 101 frames of gray-level image sequences, *Miss America* and *Salesman*, to test our motion detector. Each frame is of size 360 x 288 yielding 1620 8 x 8 boxes. To test the motion detector, we load the first two frames into two arrays. Array 1 contains the frame which is on the screen at the receiving end of the transmission, while Array 2 is the new frame. Each 8 x 8 box in the frames is tested for motion detection. If a box is classified as unchanged, the box in Array 1 is replaced by the box in Array. Once all of the boxes are tested, the next frame is loaded into Array 2, and the process is repeated. Clearly, the parameter d will influence both the compression ratios and the resulting image quality. In order to illustrate its effect on compression we have run a series of tests summarized on Table 1. In the tabulated information note that the "Total Compression Ratio" is derived from the size of the whole video sequence after motion detection, whereas the "Steady State Compression Ratio" is the average compression ratio due to motion detection over all the frames *after* the complete first frame has been transmitted. Both values do *include* the overhead due to the additional bits sent for each box of each frame: two bytes to indicate x and y indices of the block in that frame. For storage applications, a simpler and possibly more efficient scheme with one bit per block can be used: a bit value of "1" means that motion is detected in the box and that it be sent, while "0" means that the box will not be sent (and therefore that the previous frame's corresponding box should be used). However, considering network applications, we will prefer the former header so that the image transmission will not be sensitive to packet losses.

d	Miss America				Salesman			
	Compression Ratio		Frame SNR		Compression Ratio		Frame SNR	
	Total	Steady State	Min	Max	Total	Steady State	Min	Max
0.5	2.25	2.28	38.78	40.83	3.01	3.07	37.38	44.15
1.0	4.44	4.59	36.81	39.51	6.55	6.94	35.04	43.42
1.5	6.06	6.38	35.72	38.07	9.23	10.06	33.66	42.59
2.0	7.25	7.74	34.57	37.48	11.26	12.55	32.77	41.94
2.5	8.42	9.10	33.91	36.92	13.08	14.88	31.99	41.71
3.0	9.53	10.41	33.63	36.68	14.70	17.04	31.41	41.81
3.5	10.60	11.73	33.02	36.43	16.32	19.29	30.84	41.28
4.0	11.71	13.11	32.69	36.23	18.01	21.71	30.60	41.05
4.5	12.82	14.54	32.37	35.80	19.75	24.30	30.05	40.50
5.0	13.96	16.04	32.08	35.55	21.38	26.86	29.77	40.12

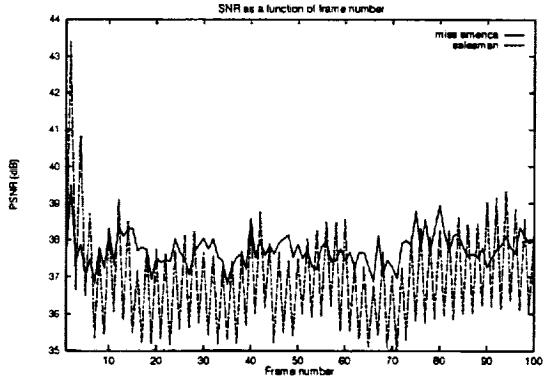
Table 1: Compression ratios obtained *only* by motion detection: as a function of difference threshold d

Other results are presented in the form of the actual images before and after motion detection. Figure 5 shows the original and the reconstructed 101st –and last– frame of the sequence with $d = 1$. In Figure 6(a), the SNR is plotted as a function of frame number for $d = 1$. Similarly Figure 6(b) shows the number of bits transmitted as a function of frame number. ¿From these results and other experiments we have run, it appears that a compression ratio of 6 or 7 can be obtained easily with a value of d close to or slightly above 1, with satisfactory image quality, when only motion detection is used for compression. In the next section this scheme will be combined with the actual neural compression of frames in order to achieve high compression ratios and satisfactory image quality.

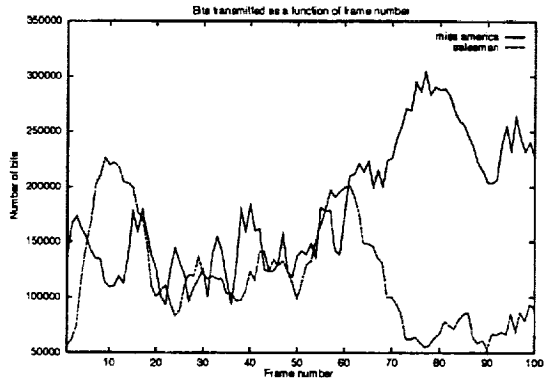
Original 101st frame

Reconstructed (SNR = 38.21)

Figure 5: Original and reconstructed last frames (101st frames) in the SALESMAN sequence using the motion detection scheme with $d = 1$



(a)



(b)

Figure 6: Experimental results for motion detection with $d = 1$: a) PSNR as a function of frame number, b) Number of bits transmitted as a function of frame number

3 Compression for Moving Images

In this section we will describe and evaluate the complete compression scheme for video sequences of natural images, using a combination of the motion detection scheme described earlier together with our adaptive still block-by-block (Figure 3) random neural network compression/decompression. Specifically, our compression scheme uses three networks:

- The first network scans successive boxes (fixed size portions of the image) in sequence, and identifies those boxes where motion has taken place, as described above. If a box is considered to be identical to the same box in the previous frame, it is not compressed or transmitted.
- The second network carries out compression of the box which is identified by the first network. In fact the second network is a set of distinct neural compression networks C_1, ... , C_L which are designed to achieve different compression levels. Each of these networks compresses the box in parallel. The choice of the compression level to be selected is carried out by the third network.
- The third network simulates the decompression, and provides a measure of the "quality" of the compression-decompression. In fact it is composed of L distinct decompression networks D_1, ... , D_L, where D_i matches C_i.

Then the pair C_i,D_i which yields the highest compression ratio at a quality level of Q or better, chosen to be acceptable for the particular application, is selected and the compressed box is transmitted. For grey-level images Q is formulated as a SNR value. Figure 7 shows the block diagram of the adaptive still image compression network. Note that with the exception of the initial learning phase, all the operations which have been outlined above can be carried out "on-the-fly", i.e. in real-time as each box goes through the transmitter, and as each compressed box goes through the receiver. (See Figure 1 for a block diagram of the total proposed scheme).

Another refinement would be to use the network D_i (which is stored both at the transmitting end and at the receiving end) to further train the network C_i in on-line mode. In this case, D_i's weights will *not* be changed, and only C_i's weights are updated.

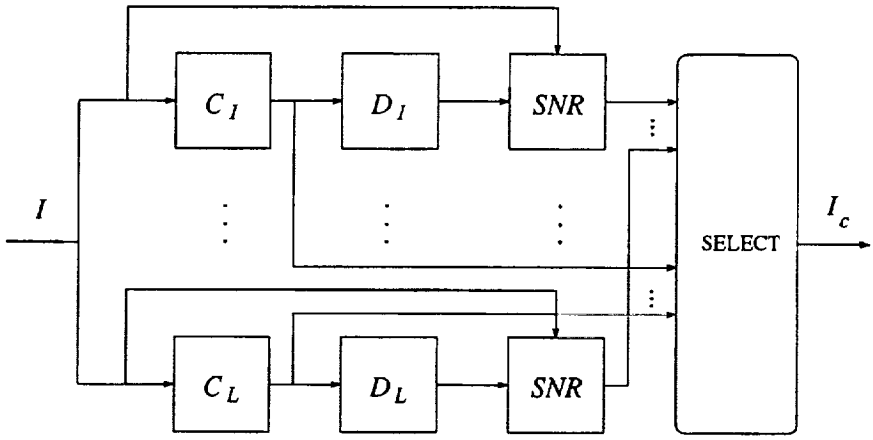


Figure 7: Block diagram of the adaptive still image compression network

At the *"receiving or decompression"* end, if the transmitter has sent a 0 bit to indicate that the current box is identical to the same box in the previous frame, then the previous frame's box is placed in the corresponding position of the output image. Otherwise the compressed box is received. Implicitly (through the box's size) or explicitly (via some variable i which would accompany the box) the compression level used is known to the receiver. We then use the network D_i to decompress the box, which is subsequently placed in appropriate sequence into the output image. The relationship between any two compression/decompression networks C_i, D_i is shown in (Figure 8).

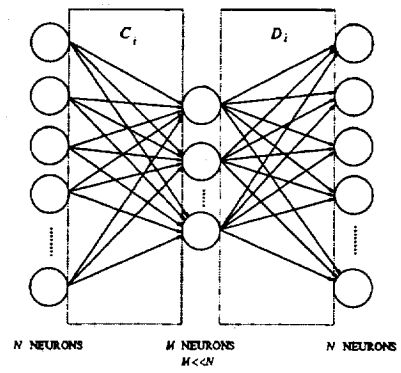


Figure 8: A Neural Network Compression/Decompression Pair

3.1 Experimental Results for Moving Image Compression

We have experimented the combined scheme with three still image compression machines ($L = 3$ with $8:1$, $16:1$ and $32:1$ compression/decompression pairs), and have tested it on the 101-frame *Miss America* and *Salesman* grey-level image sequences. Table 2 summarizes the results we have obtained for $Q = 30$.

d	Miss America				Salesman			
	Compression Ratio		Frame SNR		Compression Ratio		Frame SNR	
	Total	Steady State	Min	Max	Total	Steady State	Min	Max
0.5	21.69	27.35	31.93	33.70	21.46	31.13	26.86	31.13
1.0	32.82	48.12	32.02	34.02	36.82	57.38	28.26	35.83
1.5	38.91	62.68	32.73	34.24	45.38	81.58	28.72	37.94
2.0	42.88	73.79	32.50	34.44	50.90	101.59	28.93	38.75
2.5	46.30	84.65	32.36	34.54	55.02	119.64	28.90	38.96
3.0	48.81	95.35	32.10	34.60	58.26	136.30	28.77	39.07
3.5	51.95	105.89	32.00	34.69	61.22	153.93	28.73	39.05
4.0	54.36	116.55	31.80	34.76	63.96	172.67	28.73	39.14
4.5	56.70	128.03	31.71	34.88	66.52	192.91	28.57	39.05
5.0	58.92	140.01	31.50	34.91	68.74	213.08	28.54	39.00

Table 2: Compression ratios obtained by the combination of motion detection and still image compression with $Q = 30$: as a function of difference threshold d

In Figure 9 we show the original and the reconstructed 101st frame of *Miss America* using the complete scheme described above with $d = 1.5$ and $Q = 30$. Figure 10 indicates the variation of compression ratio over time. Figure 11 shows the running average compression ratios and the running average bits per pixel for a runlength of 1000, based on *Miss America* sequence with $d = 2$ and $Q = 30$. In Figure 12.a, PSNR is plotted as a function of frame number for $d = 2$, $Q = 30$. Figure 12.b shows the number of bits transmitted as a function of frame number.

Original 101st frame

Reconstructed (SNR = 32.83)

Figure 9: Original and reconstructed last frames (101st frames) in the Miss America sequence using the motion detection scheme with $d = 1.5$ combined with still image compression with $Q = 30$

4 Discussion and Conclusions

Many further improvements of the basic method we propose can be thought of and some are certainly worth further work. In particular the following observations can be used to design networks with enhanced

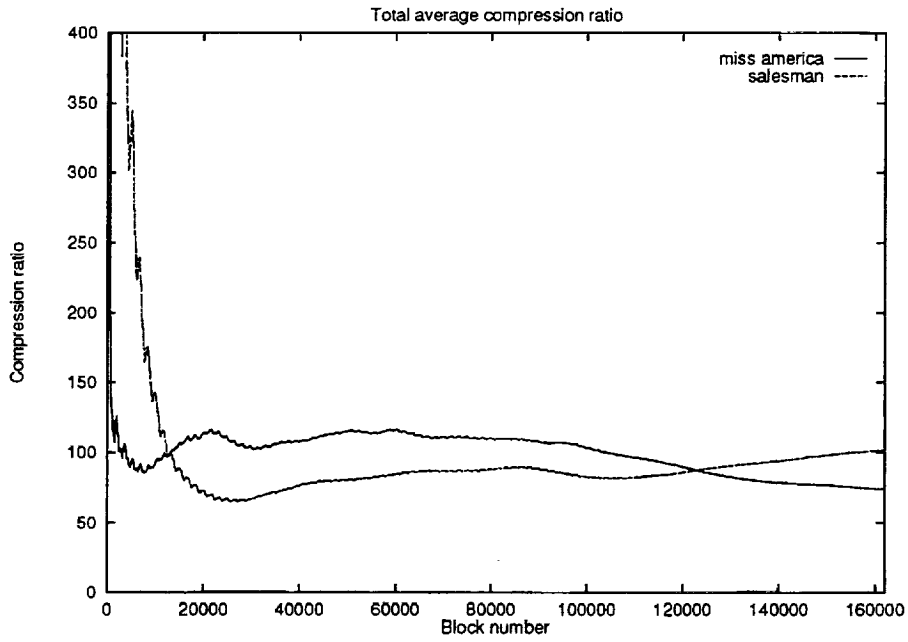
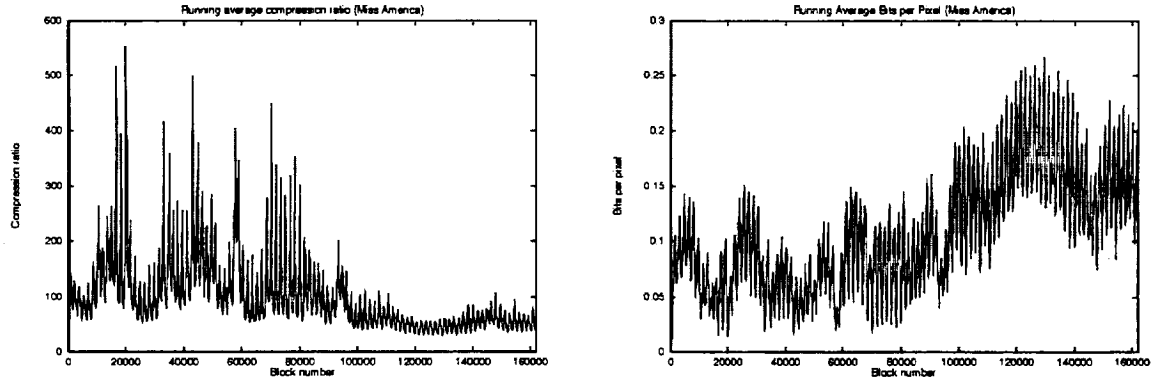


Figure 10: Total average compression ratio as a function of block number for the combined scheme with $d = 2$ and $Q = 30$



(a) (b)

Figure 11: Experimental results with MISS AMERICA sequence using the combined scheme with $d = 2$ and $Q = 30$: a) Running average compression ratio as a function of block number, b) Running average bits per pixel as a function of block number

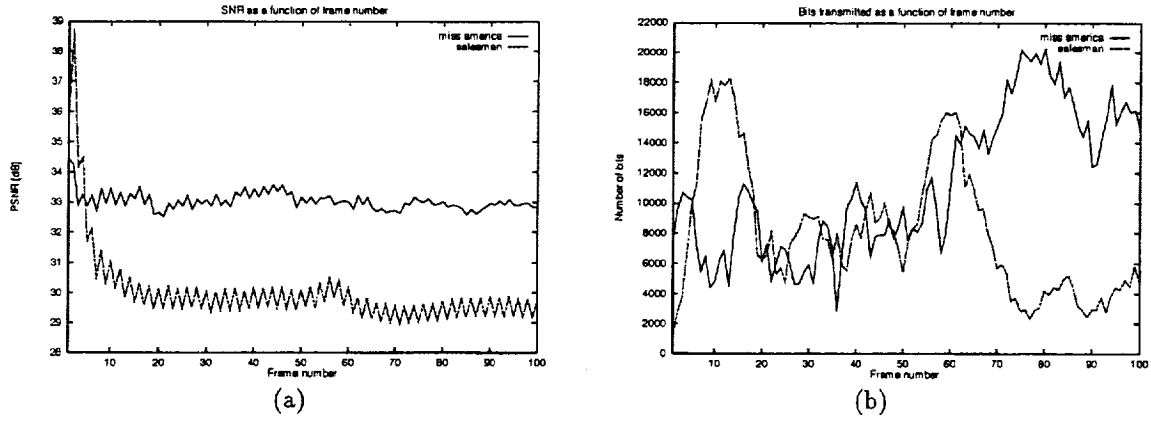


Figure 12: Experimental results for the combined scheme with $d = 2$ and $Q = 30$: a) PSNR as a function of frame number, b) Number of bits transmitted as a function of frame number

compression capabilities:

- The random neural network learning algorithm (described in the Appendix) applies to arbitrary recurrent networks. Hence, instead of restricting ourselves to fully feedforward networks, we can use feedback connections between the compressed and input layer, and the output layer and the compressed layer. Further feedback is possible and useful locally within the output layer. Such feedback can help the network find better compression/decompression parameters.
- The quality level (e.g. SNR) predicted at the transmitting end is exactly what the result is for that box, after it is decompressed at the receiver, since the networks $D_1, \ldots, D_L$ are identical both at the transmitter and receiver. Thus we propose to update the weights of the neural networks $C_1, \ldots, C_L$ constantly using gradient descent to improve performance with each individual box. This will be detrimental to the "real-time" nature of the whole approach we propose, but would be worth examining in order to obtain much higher SNR figures.
- It is also possible to store all of the compression networks $C_1, \ldots, C_L$ at the receiver – as well as at the transmitter. Then, on-going improvement via learning as compression/decompression takes place can be carried out periodically for both compression and decompression networks, at the expense of transmitting some uncompressed frames or boxes from time to time.
- Initial learning of weights can be carried out at the transmitter, or receiver, or both at the transmitter and receiver, or off-line. The resulting weights would then be loaded into the transmitter and the receiver. Note that if the sample images used for learning are known both to the transmitter and to the receiver, then the quasi-identical set of weights (to the exception of possible different numerical round-errors) can be obtained both at the transmitter and at the receiver. Thus, the images to be used as a basis for learning can be transmitted from time to time (i.e. infrequently) from one to the other in order to improve the system's compression capabilities.
- All the work described in this paper needs to be extended to colour images. Currently, learning of the weights of each C_i, D_i pair is obtained using gradient descent and the SNR ratio is used as a performance criterion is essentially equivalent to a quadratic cost function. We would use other cost metrics (such as LAB-type measures) to carry out learning for colour images.

In addition to the general scheme described above, we will examine some other enhancements related to the non-linearity of the input-output amplitude mapping of the compression/decompression scheme. We expect to obtain further quality improvement with appropriate compensation of non-linearity. This compensation

can also be part of the learning scheme. Moreover, the adaptive selection of the level of compression to be used at the transmitter side can be improved by making use of the state of the transmission medium – specifically of the network being used. This would be particularly relevant if we are dealing with an ATM (Asynchronous Transfer Mode) network. The adaptive decision can be based on feedback about network state – such as current load on the network – as well as SNR and/or visual quality metrics. For example, in case of little load on the network, we can favor small compression ratios, thus increasing visual quality. Similarly, in case of a heavily loaded network, we can sacrifice visual quality and transmit with maximal compression. This adaptive decision can also be learned.

With some of the improvements described above, we expect to achieve compression ratios better than $250:1$ for grey-level moving image sequences, and still higher levels for colour, with quality levels of the order of $SNR = 30$ for grey level images, and acceptable LAB-type measures and SNR levels for colour images.

5 Appendix: The Random Neural Network Model and its Learning Algorithm

In this appendix we provide a summary of the Random Neural Network Model and of its Learning Algorithm, in order to provide a theoretical background for the techniques which are used in this paper.

5.1 The Random Neural Network Model

In the random neural network model (Gelenbe (1989,90) [12, 13]) signals in the form of spikes of unit amplitude circulate among the neurons. Positive signals represent excitation and negative signals represent inhibition. Each neuron's state is a non-negative integer called its potential, which increases when an excitation signal arrives to it, and decreases when an inhibition signal arrives. Thus, an excitatory spike is interpreted as a "+1" signal at a receiving neuron, while an inhibitory spike is interpreted as a "−1" signal.

Neural potential also decreases when the neuron fires. Thus a neuron i emitting a spike, whether it be an excitation or an inhibition, will lose potential of one unit, going from some state whose value is k_i to the state of value $k_i - 1$.

The state of the n-neuron network at time t, is represented by the vector of non-negative integers $k(t) = (k_1(t), \ldots, k_n(t))$, where $k_i(t)$ is the potential or integer state of neuron i. We will denote by k and k_i arbitrary values of the state vector and of the i-th neuron's state.

Neuron i will "fire" (i.e. become excited and send out spikes) if its potential is *positive*. The spikes will then be sent out at a rate $r(i)$, with independent, identically and exponentially distributed inter-spike intervals. Spikes will go out to some neuron j with probability $p^+(i,j)$ as excitatory signals, or with probability $p^-(i,j)$ as inhibitory signals. A neuron may also send signals out of the network with probability $d(i)$, and $d(i) + \sum_{j=1}^{n} [p^+(i,j) + p^-(i,j)] = 1$. Let $w_{ij}^+ = r(i)\, p^+(i,j)$, and $w_{ij}^- = r(i)\, p^-(i,j)$. Here the "$w$'s" play a role similar to that of the synaptic weights in connectionist models, though they specifically represent rates of excitatory and inhibitory spike emission. They are non-negative. Exogenous (i.e. those coming from the "outside world") excitatory and inhibitory signals also arrive to neuron i at rates $\Lambda(i)$, $\lambda(i)$, respectively.

This is a "recurrent network" model, *i.e.* a network which is allowed to have feedback loops, of arbitrary topology.

Computations related to this model are based on the probability distribution of network state $p(k,t) = \Pr[k(t) = k]$, or with the marginal probability that neuron i is excited $q_i(t) = \Pr[k_i(t) > 0]$. As a consequence, the time-dependent behaviour of the model is described by an infinite system of *Chapman-Kolmogorov* equations for discrete state-space continuous Markovian systems.

Information in this model is carried by the *frequency* at which spikes travel. Thus, neuron j, if it is excited, will send spikes to neuron i at a frequency $w_{ij} = w_{ij}^+ + w_{ij}^-$. These spikes will be emitted at exponentially distributed random intervals. In turn, each neuron behaves as a non-linear *frequency demodulator* since it transforms the incoming excitatory and inhibitory spike trains' rates into an "amplitude", which is $q_i(t)$ the probability that neuron i is excited at time t. Intuitively speaking, each neuron of this model is also a frequency modulator, since neuron i sends out excitatory and inhibitory spikes at rates (or frequencies) $q_i(t)r(i)p^+(i,j)$, $q_i(t)r(i)p^-(i,j)$ to any neuron j.

The stationary probability distribution associated with the model is the quantity used throughout the com-

putations:

$$p(k) = \lim_{t\to\infty} p(k,t), \quad q_i = \lim_{t\to\infty} q_i(t), \quad i = 1,...,n. \tag{3}$$

It is given by the following result:

Theorem 1. *Let q_i denote the quantity*

$$q_i = \lambda^+(i)/[r(i) + \lambda^-(i)] \tag{4}$$

where the $\lambda^+(i), \lambda^-(i)$ for $i = 1,...,n$ satisfy the system of nonlinear simultaneous equations:

$$\lambda^+(i) = \sum_j q_j r(j) p^+(j,i) + \Lambda(i), \quad \lambda^-(i) = \sum_j q_j r(j) p^-(j,i) + \lambda(i) \tag{5}$$

Let $k(t)$ be the vector of neuron potentials at time t and $k = (k_1,...,k_n)$ be a particular value of the vector; let $p(k)$ denote the stationary probability distribution.

$$p(k) \;=\; \lim_{t\to\infty} \text{Prob}[k(t) = k]$$

If a nonnegative solution $\{\lambda^+(i), \lambda^-(i)\}$ exists to equations 4 and 5 such that each $q_i < 1$, then

$$p(k) = \prod_{i=1}^{n} [1 - q_i] q_i^{k_i} \tag{6}$$

The quantities which are most useful for computational purposes, *i.e.* the probabilities that each neuron is excited, are directly obtained from:

$$\lim_{t\to\infty} \text{Prob}[k_i(t) > 0] \;=\; q_i \;=\; \lambda^+(i)/[r(i) + \lambda^-(i)] \qquad \text{if } q_i < 1.$$

5.2 The Learning Algorithm

Let us describe the learning algorithm we use in this study. It is based on the algorithm described in (Gelenbe 93) [16].

The algorithm chooses the set of network parameters $\mathbf{W}$ in order to learn a given set of K input-output pairs $(\iota, \mathbf{Y})$ where the set of successive inputs is denoted $\iota = \{\iota_1,...,\iota_K\}$, and $\iota_k = (\Lambda_k, \lambda_k)$ are pairs of positive and negative signal flow rates entering each neuron:

$$\boldsymbol{\Lambda}_k \;=\; [\Lambda_k(1),...,\Lambda_k(n)], \qquad \boldsymbol{\lambda}_k = [\lambda_k(1),...,\lambda_k(n)]$$

The successive desired outputs are the vectors $\mathbf{Y} = \{y_1,...,y_K\}$, where each vector $y_k = (y_{1k},...,y_{nk})$, whose elements $y_{ik} \epsilon [0,1]$ correspond to the desired values of each neuron. The network approximates the set of desired output vectors in a manner that minimizes a cost function E_k:

$$E_k \;=\; \frac{1}{2}\sum_{i=1}^{n} a_i (q_i - y_{ik})^2, \quad a_i \geq 0$$

If we wish to remove some neuron j from network output, and hence from the error function, it suffices to set $a_j = 0$

Both of the n by n weight matrices $\mathbf{W}_k^+ = \{w_k^+(i,j)\}$ and $\mathbf{W}_k^- = \{w_k^-(i,j)\}$ have to be learned after each input is presented, by computing for each input $\iota_k = (\Lambda_k, \lambda_k)$, a new value $\mathbf{W}_k^+$ and $\mathbf{W}_k^-$ of the weight matrices, using gradient descent. Clearly, we seek only solutions for which all these weights are positive.

Let $w(u,v)$ denote any weight term, which would be either $w(u,v) \equiv w^-(u,v)$, or $w(u,v) \equiv w^+(u,v)$. The weights will be updated as follows:

$$w_k(u,v) = w_{k-1}(u,v) - \eta \sum_{i=1}^{n} a_i(q_{ik} - y_{ik})[\partial q_i / \partial w(u,v)]_k \tag{7}$$

where $\eta > 0$ is some constant, and

1. q_{ik} is calculated using the input ι_k and $w(u,v) = w_{k-1}(u,v)$, in equation 3.
2. $[\partial q_i / \partial w(u,v)]_k$ is evaluated at the values $q_i = q_{ik}$ and $w(u,v) = w_{k-1}(u,v)$.

To compute $[\partial q_i / \partial w(u,v)]_k$ we turn to the expression 3, from which we derive the following equation:

$$\begin{aligned} \partial q_i / \partial w(u,v) \;=\; & \sum_j \partial q_j / \partial w(u,v)[w^+(j,i) - w^-(j,i)q_i]/D(i) \\ & -\mathbf{1}[u=i]q_i/D(i) \\ & +\mathbf{1}[w(u,v) \equiv w^+(u,i)]q_u/D(i) \\ & -\mathbf{1}[w(u,v) \equiv w^-(u,i)]q_u q_i/D(i) \end{aligned}$$

Let $\mathbf{q} = (q_1, ..., q_n)$, and define the $n \times n$ matrix

$$\mathbf{W} \;=\; \{[w^+(i,j) - w^-(i,j)q_j]/D(j)\} \quad i,j = 1, ..., n$$

We can now write the vector equations:

$$\begin{aligned} \partial \mathbf{q}/\partial w^+(u,v) &= \partial \mathbf{q}/\partial w^+(u,v)\mathbf{W} + \boldsymbol{\gamma}^+(u,v)q_u \\ \partial \mathbf{q}/\partial w^-(u,v) &= \partial \mathbf{q}/\partial w^-(u,v)\mathbf{W} + \boldsymbol{\gamma}^-(u,v)q_u \end{aligned}$$

where the elements of the n-vectors $\boldsymbol{\gamma}^+(u,v) = [\gamma_1^+(u,v), \ldots, \gamma_n^+(u,v)]$, $\boldsymbol{\gamma}^-(u,v) = [\gamma_1^-(u,v), \ldots, \gamma_n^-(u,v)]$ are

$$\gamma_i^+(u,v) \;=\; \begin{cases} -1/D(i) & \text{if } u = i, v \neq i \\ +1/D(i) & \text{if } u \neq i, v = i \\ 0 & \text{for all other values of } (u,v) \end{cases}$$

$$\gamma_i^-(u,v) \;=\; \begin{cases} -(1+q_i)/D(i) & \text{if } u = i, v = i \\ -1/D(i) & \text{if } u = i, v \neq i \\ -q_i/D(i) & \text{if } u \neq i, v = i \\ 0 & \text{for all other values of } (u,v) \end{cases}$$

Notice that

$$\begin{aligned} \partial \mathbf{q}/\partial w^+(u,v) &= \boldsymbol{\gamma}^+(u,v)q_u[\mathbf{I} - \mathbf{W}]^{-1} \\ \partial \mathbf{q}/\partial w^-(u,v) &= \boldsymbol{\gamma}^-(u,v)q_u[\mathbf{I} - \mathbf{W}]^{-1} \end{aligned} \tag{8}$$

where $\mathbf{I}$ denotes the n by n identity matrix. Hence the main computational work is to obtain $[\mathbf{I} - \mathbf{W}]^{-1}$. This is of time complexity $O(n^3)$, or $O(mn^2)$ if an m-step relaxation method is used.

We now have the information to specify the complete learning algorithm for the network. We first initialize the matrices $\mathbf{W}_0^+$ and $\mathbf{W}_0^-$ in some appropriate manner. This initiation will be made at random. Choose a value of η, and then for each successive value of k, starting with $k = 1$ proceed as follows:

1. Set the input values to $\iota_k = (\Lambda_k, \lambda_k)$.
2. Solve the system of nonlinear equations 3 with these values.
3. Solve the system of linear equations (8) with the results of (2).
4. Using equation 7 and the results of (2) and (3), update the matrices $\mathbf{W}_k^+$ and $\mathbf{W}_k^-$. Since we seek the "best" matrices (in terms of gradient descent of the quadratic cost function) that satisfy the *nonnegativity* constraint, in any step k of the algorithm, if the iteration yields a negative value of a term, we have two alternatives:
   (a) Set the term to zero, and stop the iteration for this term in this term in this step k; in the next stop $k+1$ we will iterate on this term with the same rule starting from its current null value;
   (b) Go back to the previous value of the term and iterate with a smaller value of η.

This general scheme can be specialized to feedforward networks yielding a computational complexity of $O(n^2)$, rather than $O(n^3)$, for each gradient iteration.

References

[1] Adelson, E.H., Simoncelli, E. "Orthogonal pyramid transforms for image coding", *Visual Communications and Image Processing II*, Proc. SPIE, Vol.845, pp.50–58, 1987.

[2] Anthony D. "A comparison of image compression by a Neural Network and Principle Component Analysis". *Proc. International Joint Conference on Neural Networks (IJCNN'90)*, pp. 339–344. IEEE, 1990.

[3] Atalay V., Gelenbe E., Yalabık N., "The random neural network model for texture generation", *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 6, No. 1, pp 131-141, 1992.

[4] Atalay V., Gelenbe E., "Parallel algorithm for colour texture generation using the random neural network model", *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 6, No. 2-3, pp 437-446, 1992.

[5] Carrato, S., Marsi, S. "Parallel Structure Based on Neural Networks for Image Compression", *Electronics Letters*, Vol.28, No.12, pp. 1152–1153, June 1992.

[6] Carrato, S. "Neural networks for image compression," in Gelenbe, E. (ed.) *"Neural Networks: Advances and Applications 2"*, Elsevier North-Holland, pp. 177-198, 1992.

[7] Chiang Y.W. "Motion estimation using a neural network." *Proc. IEEE International Symposium on Circuits and Systems*. IEEE, 1990.

[8] Cottrell, G.W., Munro, P., Zipser, D. "Image compression by backpropagation: an example of extensional programming,", in Sharkey, N.E., (ed.) *Models of cognition: a review of cognition science*, NJ:Norwood, 1989.

[9] Courellis S.H. "An Artifical Neural Network for Motion Detection and Speed Estimation". *Proc. International Joint Conference on Neural Networks (IJCNN'90*, pp. 407–421. IEEE, 1990.

[10] Daugman J.G. "Relaxation Neural Network for Non-Orthogonal Image Transformations' '. *Proc. International Conference on Neural Networks*. IEEE, 1988.

[11] Feng W.C. "Real-Time Neuroprocessor for Adaptive Image Compression Based upon Frequency-Sensitive Competetive Learning". *Proc. The International Joint Conference on Neural Networks (IJCNN'91)*, pp 429. IEEE, 1991.

[12] Gelenbe E., "Random neural networks with negative and positive signals and product form solution", *Neural Computation*, Vol. 1, No. 4, pp 502-511, 1989.

[13] Gelenbe E., "Stability of the random neural network model", *Neural Computation*, Vol. 2, No. 2, pp. 239-247, 1990.

[14] Gelenbe, E., Stafylopatis, A., "Global behaviour of homogeneous random neural systems", *Applied Math. Modelling*, **15** (1991), pp. 535–541.

[15] Gelenbe E., Stafylopatis A., Likas A., "An extended random network model with associative memory capabilities", *Proc. International Conference on Artificial Neural Networks (ICANN'91)*, Helsinki, June 1991.

[16] Gelenbe E., "Learning in the recurrent random neural network", *Neural Computation*, Vol. 5, No. 1, pp 154-164, 1993.

[17] Gelenbe, E., Sungur, M. "Image compression with the random neural network", to appear in *Proc. International Conference on Artificial Neural Networks*, North-Holland Elsevier, 1994.

[18] Gray, R.M. "Vector Quantization", *IEEE ASSP Magazine*, Vol.1, No.2, pp.4–29, April 1984.

[19] Huang S.J. "Image Data Compression and Generalization Capabilities of Backpropagation and Recirculation Networks". *Proc. International Symposium on Circuits and Systems*, page 1613. IEEE, 1991.

[20] Jacquin, A.E. "Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations", Vol.1, No.1, p.18–30, January 1992.

[21] Klein S.A. "'Perfect' Displays and 'Perfect' Image Compressionin Space and Time". *Human Vision, Visual Processing and Digital Display*, pp. 190–205. SPIE, 1991.

[22] Kohno R. "Image compression using a neural network with learning capability of variable function of the neural unit." *Visual Communication and Image Processing*, pp. 69–75. SPIE, 1990.

[23] Kohonen, T. *Self Organization and Associative Memory*, Springer-Verlag:Berlin, 1989.

[24] Kunt, M., Benard, M., Leonardi, R. "Recent Results in High-Compression Image Coding", *IEEE Transactions on Circuits and Systems*, Vol.CAS–34, No.1, pp. 1306–1336, 1987.

[25] LeGall, D. "MPEG : A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, Vol. 34, No. 4, pp. 46–58, April 1991.

[26] Marsi S. "Improved Neural Structures for Image Compression". *Proc. International Conference on Acoustic Speech and Signal Processing (ICASSP'91)*, page 2821. IEEE, 1991.

[27] Martine Naillon. *Advances in Neural Processing Systems*. Morgan-Kaufmann, 1989.

[28] Namphol A. "Higher Order Data Compression with Neural Networks". *Proc. The International Joint Conference on Neural Networks (IJCNN'91)*, pp. 55–59. IEEE, 1991.

[29] Nasrabadi N.M. "Vector quantization of images based upon Kohonen self organizing feature maps." *Proc. International Conference on Neural Networks*. IEEE, 1988.

[30] Ramamurthi, B., Gersho, A., "Classified Vector Quantization of Images", *IEEE Transactions on Communications*, Vol.COM–34, No.11, pp.1105–1115, November 1986.

[31] Rumelhart, D.E., McClelland, J.L. and the PDP Research Group (1986) *"Parallel Distributed Processing"*, Volumes 1 & 2, MIT Press, 1986.

[32] Sonehara N. "Image Data Compression Using a Neural Network Model". *Proc. International Joint Conference on Neural Networks (IJCNN'89)*. IEEE, 1989.

[33] Storer, J.A. (1988) *"Data Compression: Methods and Theory"*, Computer Science Press, Rockville, MD, 1988.

[34] Wallace, G.K., "The JPEG Still Picture Compression Standard, *Communications of the ACM*, Vol. 34, No. 4, pp. 30–44, April 1991.

[35] Woods, J., O'neil, S.D. "Subband Coding of Images", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol.ASSP–34, No.5, pp.1278–1288, October 1986.

[36] Zettler, W., Huffman, J., Linden, D.C.P. "Application of Compactly Supported Wavelets to Image Compression", *Image Processing Algorithms and Techniques*, Proc. SPIE, Vol.1244, pp.150–160, 1990.