Traffic and Video Quality with Adaptive Neural Compression

Erol Gelenbe, Mert Sungur *, Chris Cramer, Pamir Gelenbe
Department of Electrical Engineering
Duke University
Durham, N.C. 27708-0291
Fax: 919-660 5293, E-mail: erol@ee.duke.edu

August 8, 1994

Abstract

Video sequences are major sources of traffic for Broadband ISDN networks, and video compression is fundamental to the efficient usage of such networks. In this paper we present a novel neural method to achieve real-time adaptive compression of video sources which will tend to maintain a target quality of the decompressed image, which is specified by the user. The method uses a set of compression/decompression neural networks of different levels of compression, as well as a simple motion detection procedure. We describe the method and present experimental data concerning its performance and traffic characteristics with real video sequences. The impact of this compression method on ATM cell traffic is also investigated and measurement data is provided.

1 Introduction

Sources of real-time traffic are often highly unpredicatable with respect to the instantaneous and average load which they create. Yet such sources will provide the majority of traffic in future ATM networks, and will also necessarily affect existing datagram networks. One major source of such traffic originates in video which needs to be compressed in some form. Modern video compression techniques generate variable bit rates, since they take advantage of motion in the scenes. Therefore it becomes of great interest to relate the compression method to the traffic that it generates in the network. Such information can be used in many ways: it can be used for traffic modeling and prediction of quality of service, and it can also be used to design adaptive compression algorithms which meet constraints on the traffic or on the quality of service for users. In the latter case, it is important to note that quality of service for video users should relate not only to issues such as cell loss rates, delay and jitter, but also to the visual quality of the received decompressed video sequence.

The use of feedback from the network for the control of incoming traffic has been examined by various authors (see for example [14]). The principle of being able to vary video bit rates

in response to network conditions is not new and several authors have recently addressed this intriguing issue [15, 13, 16, 17, 41, 42]. In particular in [5] a scheme which modifies the parameters of a video coder in response to changing conditions in the Internet has been presented and tested in the H.261 coder of the IVS videoconferencing system [40]. However, we do not know of schemes which will vary compression ratios so as to meet certain levels of quality of the decompressed image.

In this paper we describe a software video compression and decompression scheme based on a neural algorithm using our pulsed "random neural network" model [18, 22]. The method we propose uses simple motion detection to determine whether a portion of the image needs to be transmitted. If transmission is needed, then a set of learning neural networks are used for compression and decompression. The level of compression is adaptively chosen so as to meet an image quality level Q which is specified by the user. The sensitivity d of the motion detector can also be varied to modify compression levels and resulting image quality. Our method is very fast and has been implemented for real-time operation in software.

In the following sections we survey the literature in the area, then present our method in detail. We test it on two commonly available video sequences, and measure the resulting bit rates and image quality. We also look at the ATM traffic which would result from using our method with these real video sequences and measure certain of its characteristics.

1.1 Compression of moving images

Lossless compression is adequate when low compression ratios are acceptable. However, substantial compression ratios can only be achieved with *lossy* compression schemes. 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.

A simplified schematic representation of a moving image compression method is shown in Figure 1.

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

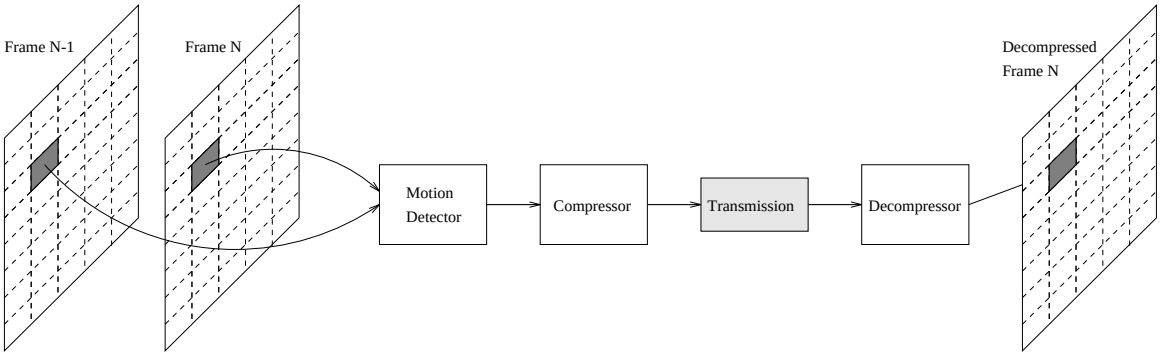


Figure 1: Block Diagram of a Video compression scheme.

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).

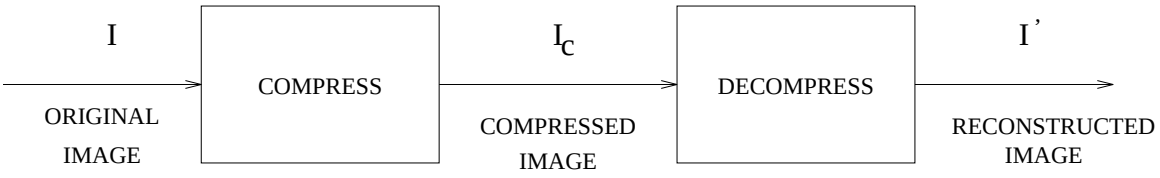


Figure 2: Image Compression Block Diagram

In *lossy compression*, the peak signal-to-noise ratio (SNR) is often used as the measure of similarity or of dissimilarity, although it does not always reflect perceived visual quality as well as one would like. For moving images, the compression ratio may vary dynamically with the specific image or image portion being transmitted, since advantage will be taken of the existence or non-existence of significant motion in successive image frames. However the SNR metric can still be used to compare corresponding frames in the original and decompressed image sequence.

Let the original and reconstructed images be denoted by functions $f(x, y)$ and $g(x, y)$ of the pixel plane position (x, y), respectively. The SNR for the reconstructed image $g(x, y)$ is defined by:

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

where:

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

1.2 Previous work

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, respectively, JPEG [43] and MPEG [31]. These schemes provide high compression ratios with good picture reconstruction qualities. The amount of computation required for both is generally high for real-time applications, so that they must be implemented in hardware. 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. 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 blocks, 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 [44], 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 [24] 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 ([36]), the computation overhead still exists. Recently, novel approaches have been introduced based on pyramidal structures [1], wavelet transforms [45], and fractal transforms [26]. These and some other new techniques [30] 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 [37] 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 [29] 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 [33]. A Kohonen net method for codebook compression is demonstrated in [35]; it seems to perform slightly better than another standard method of generating codebooks. Cottrell et al. ([9]) 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 [11], 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 [32]. It uses 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 [38]. In [25], 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 [12], the authors present a VLSI implementation of a neuro vector quantization/codebook algorithm. In [28], 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. In [34], 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 earlier results for the compression of previously "unseen" still images provide substantially better quality, especially at the lower compression ratios (8:1 and 16:1) [23] where we obtain a SNR close to 30db for a 16:1 ratio.

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 [10], 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 [8], 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 Moving image compression with the random network

One of the common neural approaches in image compression is to train a network to encode and decode the input data [9], 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 the still image approach we describe in [23] both the input, intermediate and output images are subdivided into equal-sized blocks and compression is carried out block by block (see Figure 3), as in JPEG and MPEG. This has the desirable effect of reducing the neural network learning time. It also achieves good generalization, since the numerous blocks, which make up the test image used for learning, are used as the training set. The amount of information representing the compression and decompression algorithm (i.e. the "neural network 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.

Specifically we use 8×8 pixel boxes and 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).

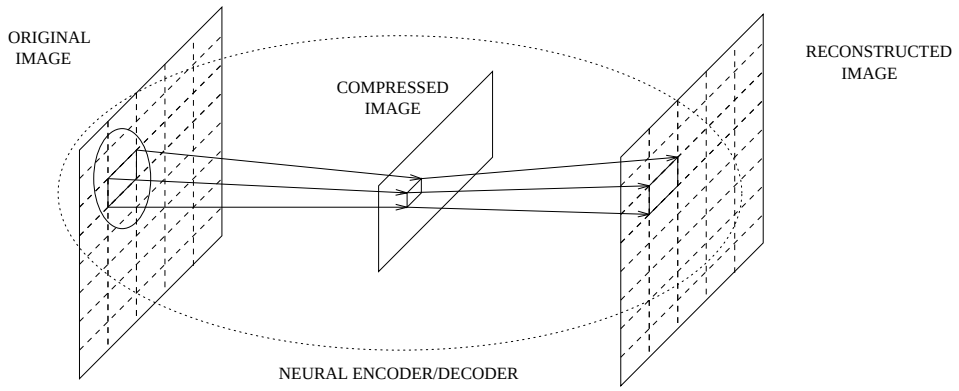


Figure 3: Compression of an arbitrarily large image using a neural encoder/decoder

The network 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 [22] 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 [23].

PEPPERS original SNR = 27.82

Figure 4: $16:1$ still image compression (0.5 bits/pixel) with random neural network

2.1 Motion detection

Since we deal with 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 do move relative to the background, but this displacement may be quite small between any two successive frames. We use this in order to perform motion detection.

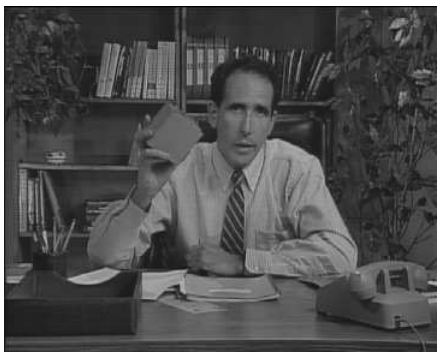
Specifically we examine the 8×8 boxes from successive frames F_{i-1}, F_i. Motion is sensed if the average grayscale value of a box in F_i differs from that of the corresponding box in frame F_{i-1} by more than a certain amount d. We have observed experimentally that the difference in the average grayscale 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.

In the data presented below, we use the gray-level image sequences *Miss America* and *Salesman* to test our motion detector. Each frame is of size 360×288 pixels, yielding 1620 8×8 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 8×8 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, 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 two bytes to indicate the x and y indices of each block in a frame.

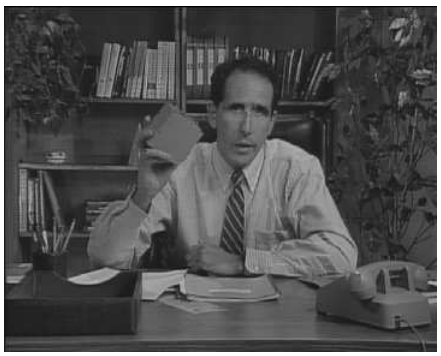
Other results are presented in the form of the actual images before and after motion detection. Figure 5 shows the original and the reconstructed 101st frame of the sequence with $d = 1$. In Figure 6(a), the SNR is plotted as a function of frame number for $d = 1$. Similarly Figure 6(b) shows the number 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.

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 d



Original 101st frame



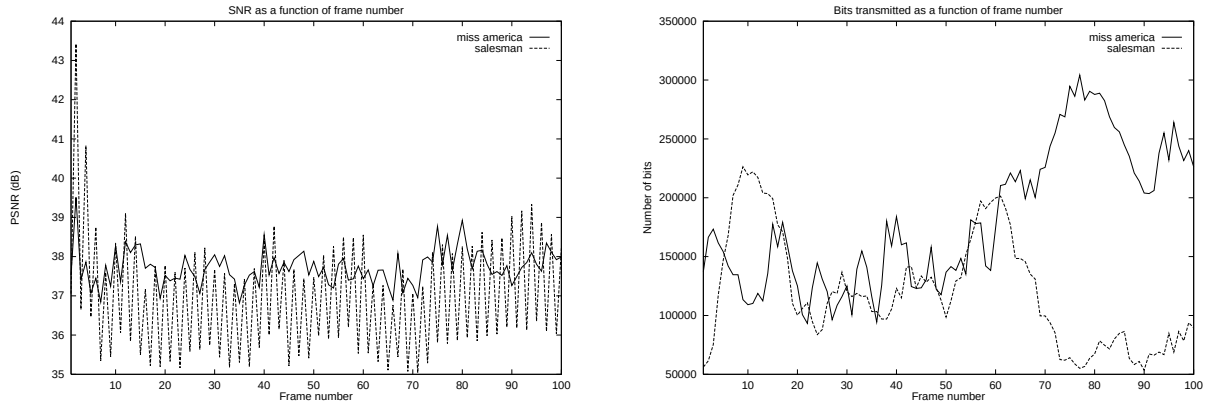
Reconstructed (SNR = 38.21)

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

2.2 Compression for Moving Images

We will now 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 technique. 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 $C_1, \ldots, C_L$ 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



(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

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 $D_1, \ldots, D_L$, where D_i matches C_i.

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 learning phase which is used to train all of these networks using an unseen image (in our case the well-known "Lena" image), all the operations which have been outlined above are carried out "on-the-fly", i.e. in real-time as each box leaves the sender, and as each compressed box goes into the receiver and is decompressed (see Figure 1). The relationship between any two compression/decompression networks C_i, D_i is shown in Figure 8.

Another refinement, which we have not tested, 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.

At the *"receiving or decompression"* end, if the transmitter has indicated 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 a variable i which accompanies the box) the compression level used is known to the receiver. Therefore the appropriate network D_i is used to decompress the box, which is then placed in sequence into the output image.

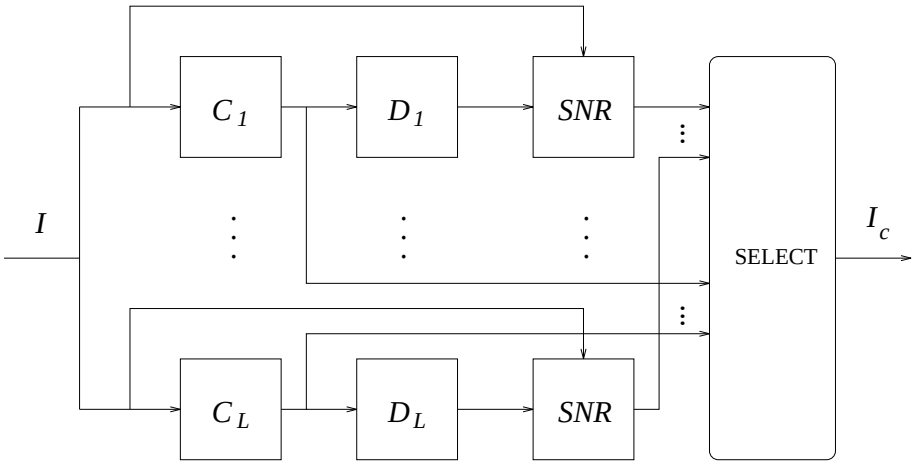


Figure 7: Block diagram of the adaptive still image compression network

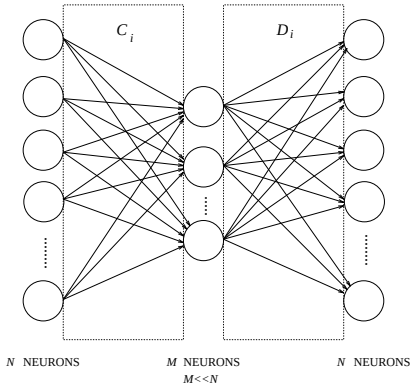


Figure 8: Neural network compression/decompression pair

3 Experimental Results

The experimental results we now present are obtained using three compression/decompression pairs ($L = 3$) with $8:1$, $16:1$ and $32:1$ compression ratios. The target decompressed image quality is set to the SNR value of $Q = 30$. The grey-level video sequences used in the tests are *Miss America* and *Salesman* which were mentioned previously.

In Table 2 we summarize the experimental results for different values of the motion detection threshold d. In each case, d is fixed and the same video sequences are presented as input to the compression/decompression software.

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 motion detection and compression with $Q = 30$ as a function of 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, the SNR 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.

All of these results confirm the effectiveness of the method we propose in obtaining relatively high compression ratios with good image quality. They also illustrate the fact that our compression method will provide time varying traffic, and that it will strongly depend on the specific image sequence which is being transmitted. Although we only report results for two video sequences, we have also tested our method for other well-known sequences (such as the Ping-Pong Player).

Original 101st frame

Reconstructed (SNR = 32.83)

Figure 9: Original and reconstructed 101st frames in the MISS AMERICA sequence with $d = 1.5$ and $Q = 30$

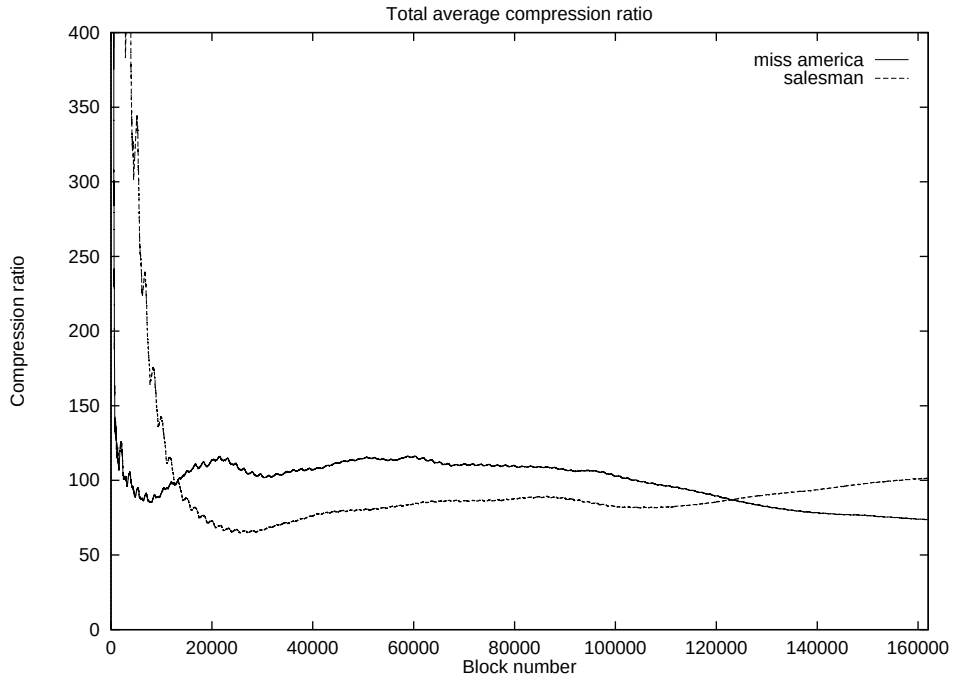
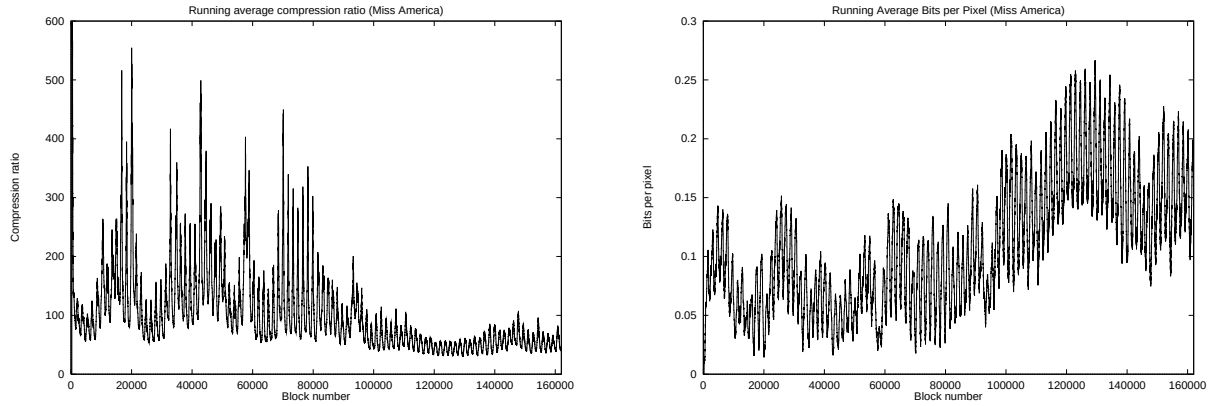
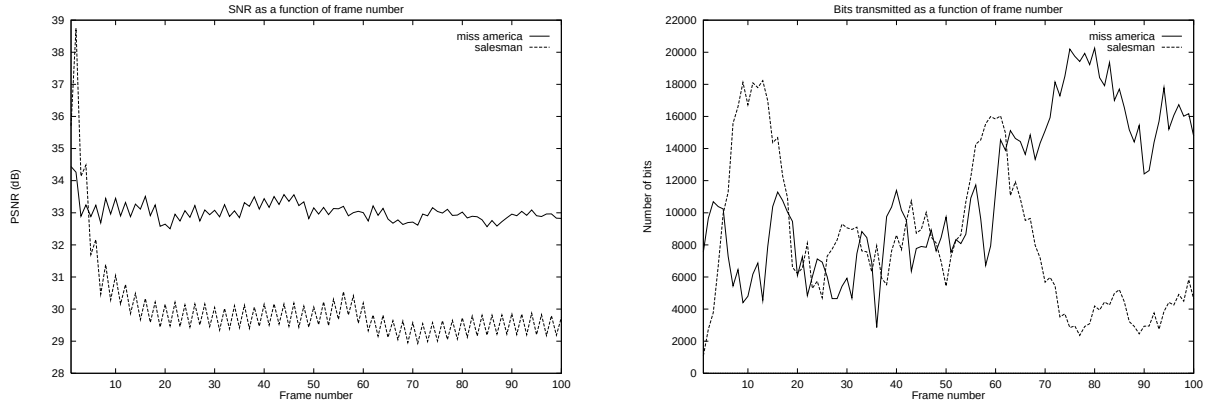


Figure 10: Total average compression ratio as a function of block number with $d = 2$ and $Q = 30$



(a) (b)

Figure 11: Experimental results for MISS AMERICA sequence 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



(a) (b)

Figure 12: Experimental results 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

3.1 Simulating ATM traffic from the compressed video sources

We now turn to a study of the traffic which is being generated by our method. Since the video sequences produce images at a rate of 30 frames/second, and because each transmitted frame is composed either of compressed blocks, or of positional information concerning a block which does not need to be transmitted, it is easy to translate the output of the compression system into either an instantaneous rate of bits or ATM cells transmitted per time unit. In the case of ATM cells, we assume that compressed blocks are placed into 58-byte ATM cells so that any one block cannot span two different cells.

Thus on Figure () we show the traffic rate in ATM cells/second which is being generated for the *Miss America* video sequence over a few seconds. We observe the highly unpredictable nature of the traffic and it's time-varying behaviour. Figure () presents the corresponding autocorrelation function of cell traffic generated by our compression method for $d = 0.5$ and $d = 2$, indicating a linear decrease over a few seconds. These results are presented for the same target image quality $Q = 30$.

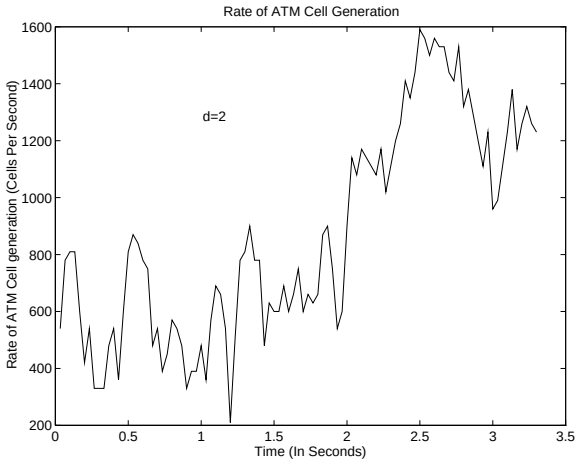
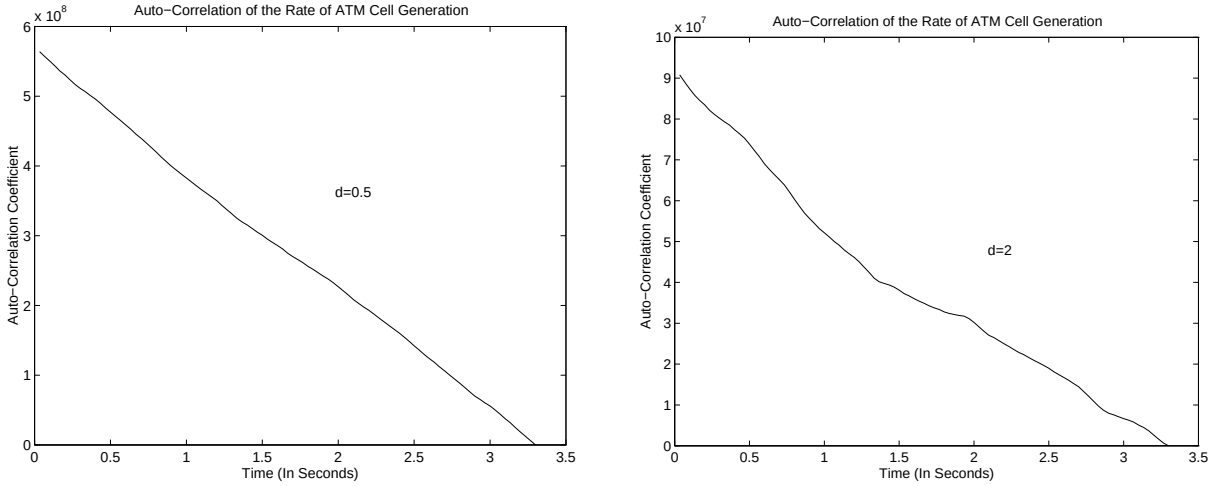


Figure 13: ATM traffic rate with $d = 2$

Much ATM traffic can be expected to travel over relatively slow legacy networks for a number of years to come, especially for portions of the network which are geographically close to the user generating the video traffic. Therefore we also examine the behaviour of a finite ATM buffer queue for these traffic streams, and measure the cell loss rate for different values of d, different buffer sizes, and different speeds at which the buffer is being emptied. These results are summarized on Figures () and () for the *Miss America* sequence. We have purposely chosen a slow legacy network speed of 64 Kbytes/sec to evaluate the losses observed on a link when a link capacity which is currently quite realistic is used. We see that choosing a larger value of d can substantially reduce the loss rate even for small buffers, and that the compression scheme makes a very substantial difference in the cell loss rate even when the network is slow and the



(a) (b)

Figure 14: Autocorrelation function of ATM cell traffic with (a) $d = 0.5$ and (b) $d = 2$

buffer sizes are small. In order to see what the effect of doubling the link speed can be, on Figure () we plot the ATM cell loss rate for a 128 Kbyte/sec with the same video sequence and with $d = 3$; clearly by comparing results with Figure Figure (), it is clear that a substantial reduction is cell loss can be achieved by doubling link speeds even when link speeds are relatively small.

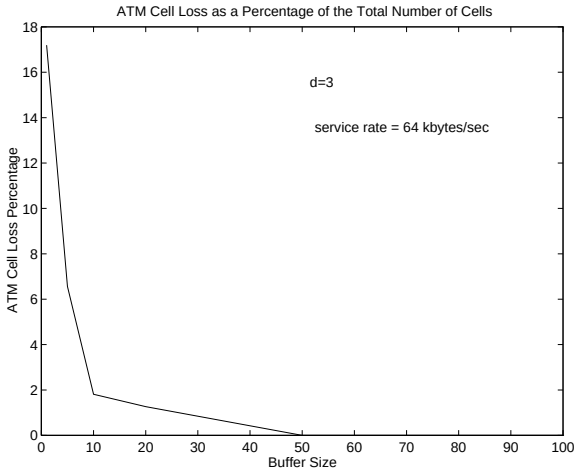


Figure 15: Percentage of lost cells vs buffer size for a 64 Kbyte/sec link with $d = 3$

4 Conclusions

In this paper we have descibed an adaptive neural technique for video compression and have studied its traffic charcateristics. Our compression method is designed to meet quality requirements with respect to the decompressed image at the receiver. It can vary the traffic rate it generates into the network by modifying a simple parameter which is used to detect motion in

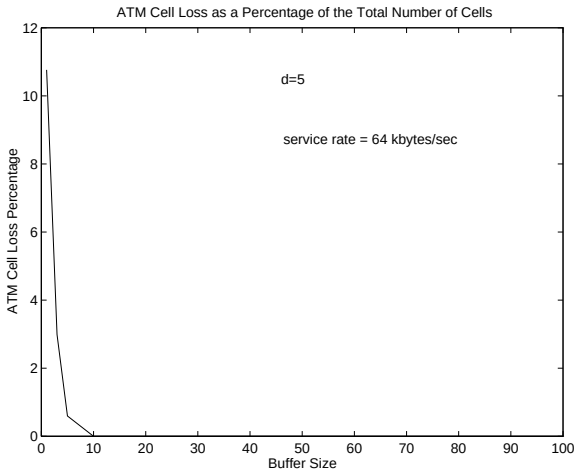


Figure 16: Percentage of lost cells vs buffer size for a 64 Kbyte/sec link with $d = 5$

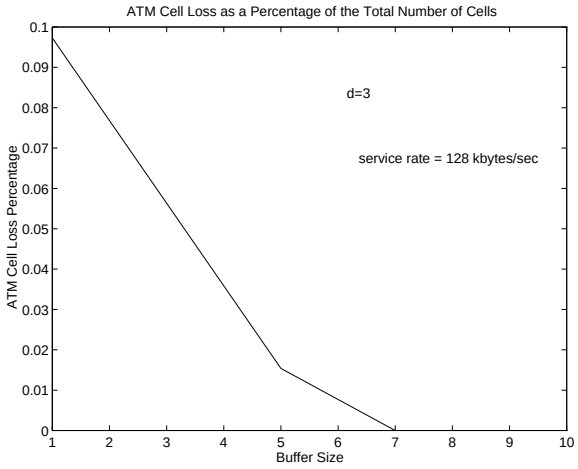


Figure 17: Percentage of lost cells vs buffer size for a 128 Kbyte/sec link with $d = 3$

successive frames, or it can do this by modifying the target image quality requirement.

The presention of these ideas and algorithms is complemented by a substantial amount of experimental data concerning effective compression ratios and resulting image quality.

We also present experimental data concerning the ATM traffic that our method will generate, including measurement of cell traffic rates, autocorrelation of the traffic and buffer overflow for finite sized ATM buffers. These results are based on real image sequences which are used to drive our moving image compresson method.

Many further improvements of the basic method investigated in this paper can be envisoned, and some are certainly worth further work. In particular we can adaptively vary the motion detection threshold in real-time to meet a combined image quality and effective bit rate requirement. We can also introduce on-line learning of certain parameters at the transmitter to continuously improve the ability of the system to compress moving images as image characteristics change with time. Many of these improvements will lead to greater computational complexity and/or the need to exchange more information between the sender and receiver, so that any extension of the method needs to be carefully evaluated in terms of the major trade-offs which are addressed.

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

In this appendix we summarize the Random Neural Network Model and of its Learning Algorithm. In the random neural network model (Gelenbe (1989,90) [18, 19]) 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 computations:

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

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:

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

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.

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

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:

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

Let us now describe the learning algorithm we use in this study. It is based on the algorithm described in (Gelenbe 93) [22].

The algorithm chooses the set of network parameters $\mathbf{W}$ in order to learn a given set of K input-output pairs ($\boldsymbol{\iota}$, $\mathbf{Y}$) where the set of successive inputs is denoted $\boldsymbol{\iota} = \{\iota_1, ..., \iota_K\}$, 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:

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

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] Bolot, J.-C., Turletti, T. "A rate control mechanism for packet video in the Internet", *Proc. IEEE INFOCOM'94*, Vol. 3, pp 1216-1223, Toronto, June 1993.

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

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

[9] 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.

[10] 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.

[11] Daugman J.G. "Relaxation neural network for non-orthogonal image transformations' '. *Proc. International Conference on Neural Networks*. IEEE, 1988.

[12] 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.

[13] Chen, C.-T., Wong, A. "A self-governing rate buffer control strategy for pseudoconstant bit rate video encoding", *IEEE Transactions on Image Processing*, Vol. 2, No. 1, pp 50-59, January 1993.

[14] Fendick, K., Rodriguez, M., Weiss, A. "Analysis of a rate based control strategy with delayed feedback", *Proc. ACM SIGCOMM'92*, Baltimore, pp 136-142, 1992.

[15] Gilge, M., Gusella, R. "Motion video coding for packet-switching networks", *Proc. SPIE Conference on Visual Communications and Image Processing*, Boston, November 1991.

[16] Jeffay, K., Stone, D.L., Talley, T., Smith, D. "Motion video coding for packet-switching networks", *Proc. 3rd Int. Workshop on Network and OS Support for Digital Audio and Video*, San Diego, November 1992.

[17] Kanakia, H., Mishra, P., Reibman, A. "An adaptive congestion control scheme for real-time video transport", *Proc. ACM SIGCOMM'93*, San Francisco, pp 20-31, September 1993.

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

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

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

[21] 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.

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

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

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

[25] 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.

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

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

[28] 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.

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

[30] 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.

[31] LeGall, D. "MPEG : A video compression standard for multimedia applications. *Communications of the ACM*, Vol. 34, No. 4, pp. 46–58, April 1991.

[32] Marsi S. "Improved neural structures for image compression". *Proc. International Conference on Acoustic Speech and Signal Processing (ICASSP'91)*, page 2821. IEEE, 1991.

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

[34] Namphol A. "Higher Order data compression with neural networks". *Proc. The International Joint Conference on Neural Networks (IJCNN'91)*, pp. 55–59. IEEE, 1991.

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

[36] Ramamurthi, B., Gersho, A., "Classified vector quantization of images", *IEEE Transactions on Communications*, Vol.COM–34, No.11, pp.1105–1115, November 1986.

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

[38] Sonehara N. "Image data compression using a neural network model". *Proc. International Joint Conference on Neural Networks (IJCNN'89).* IEEE, 1989.

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

[40] Turletti, T. "H. 261 software coder for videoconferencing over the Internet", *INRIA Research Report 1834*, January 1993.

[41] Wakeman, L. "A combined admission and congestion control scheme for variable bit-rate video", *UCL Technical Report*, University College London, June 1993.

[42] Wakeman, L. "Packetized video: Options for interaction between the user, the network and the codec", *The Computer Journal*, Vol. 36, No. 1, 1993.

[43] Wallace, G.K., "The JPEG Still picture compression standard, *Communications of the ACM*, Vol. 34, No. 4, pp. 30–44, April 1991.

[44] 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.

[45] 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.