Multimedia Systems (1996) 4: 357–369

Multimedia Systems

Traffic and video quality with adaptive neural compression

Erol Gelenbe, Mert Sungur[1], Christopher Cramer, and Pamir Gelenbe[2]

Department of Electrical Engineering, Duke University, Durham, NC 27708-0291, USA

Abstract. Video sequences are major sources of traffic for broadband ISDN networks, and video compression is fundamental to the efficient use of such networks. We present a novel neural method to achieve real-time adaptive compression of video. This tends to maintain a target quality of the decompressed image 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 are provided.

Key words: Compression/decompression neural networks – Motion detection – ATM traffic

1 Introduction

Sources of real-time traffic are often very unpredicatable with respect to the instantaneous and average load that 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 that must be compressed in some form. Modern video compression techniques generate variable bit rates, since they take advantage of motion in the scenes. Therefore, it is 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 that 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, Fendick et al. (1992)]. 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 (Chen and Wong 1993; Gilge and Gusella 1991; Jeffay et al. 1992; Kanakia et al. 1989; Wakeman 1993a,b). In particular, Bolot and Turletti (1993) present a scheme that modifies the parameters of a video coder in response to changing conditions in the Internet. It was tested in the H.261 coder of a videoconferencing system Turletti (1993). However, we do not know of schemes that vary compression ratios so as to meet certain levels of quality of the decompressed image.

In this paper we describe a scheme for software-video compression and decompression based on a neural algorithm that uses our pulsed "random neural network" model (Gelenbe 1989, 1993). 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 the 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 that 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

Present addresses:
[1]Department of Electrical and Electronics Engineering, Middle East Technical University, Ankara, Turkey
[2]Ecole Centrale des Arts et Manufactures, Paris, France
Correspondence to: E. Gelenbe
e-mail: erol@ee.duke.edu

can only be achieved with *lossy* compression schemes. The aim of image compression is to encode images or image sequences into as few bits as possible with a decoding mechanism that 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, and in those in which the source produces data at a very high rate. It is therefore often important to be able to compress and decompress "on the fly" without additional delay in conveying the image.

A simplified schematic representation of a method for moving image compression is shown in Fig. 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 so that the image I' reconstructed by decoding from this reduced representation is as similar to the original image as possible. 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 (Fig. 2).

In *lossy compression*, the peak signal-to-noise ratio (PSNR) 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 is taken of the existence or nonexistence of significant motion in successive image frames. However, the PSNR metric can still be used to compare corresponding frames in the original and decompressed image sequences.

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 PSNR for the reconstructed image $g(x, y)$ is defined by:

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

where:

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

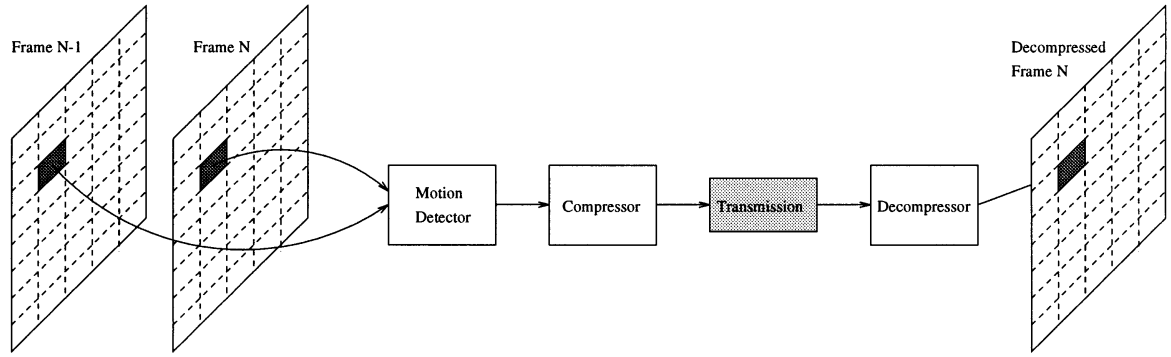
1.2 Previous work

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 (Wallace 1991) and MPEG (LeGall 1991). These schemes provide large compression ratios with good picture-reconstruction qualities. The amount of computation required for both is generally large 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, which gives an automatic 2:1 compression ratio, (2) JPEG encoding based on the discrete cosine transform (DCT) and quantization followed by some lossless compression, which yields compression ratios as large 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.

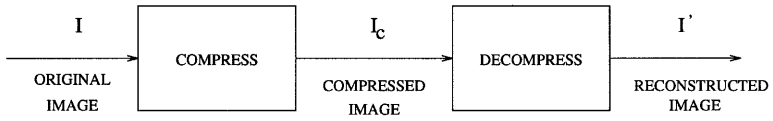
Two classical techniques for still image compression are transform and subband encoding. In transform coding techniques, the image is subdivided into small blocks, each of which undergoes a reversible linear transformation (Fourier, Hadamard, Karhunen-Loeve, etc.), followed by quantization and coding to redue redundant information in the transformed domain. In subband coding (Woods and O'Neill 1980) 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 (Gray 1984), which can achieve large 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 suffers mainly from edge degradation and great computational complexity. Although more sophisticated vector quantization schemes have been proposed to reduce edge effects (Ramamarthi and Gersha 1986), the computation overhead still exists. Recently, novel approaches have been introduced on the basis of pyramidal structures (Adelson and Simoncelli 1987), wavelet transforms (Zettler et al. 1990), and fractal transforms (Jacquin 1992). These and some other new techniques (Kunt et al. 1987) inspired by the representation of visual information in the brain, can achieve large 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, and videophones, which are all likely to be a part of daily life in the near future. Artificial neural networks (Rumelhart et al. 1986) 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 effectively be used for image compression.

Several researchers have used the Learning Vector Quantization (LVQ) network (Kohonen 1987) for developing codebooks with a distribution of codewords that approximates the probabilistic distribution of the data to be presented. A Hopfield network for vector quantization that achieves a compression of less than 4:1 is reported in (Naillon 1989). Nasrabadi (1988) demonstrates a Kohonen net method for codebook compression. It seems to perform slightly better than other standard methods of generating codebooks. Cottrell et al. (1989) 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



1



2

Fig. 1. Block diagram of a video compression scheme

Fig. 2. Block diagram of image compression

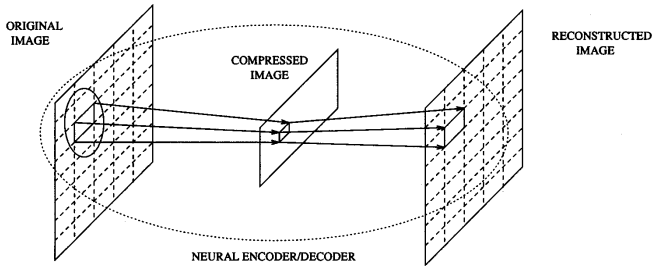
many researchers such as (Carrato 1992). Daugman (1988) presents a neural network method for finding coefficients of a 2D Gabor transform. This two-way function can then be quantized and encoded to give good images at a compression of under 1 bit/pixel, and as low as 0.38 bits/pixel with good image quality in a particular case.

A feed-forward neural network model to achieve a 16:1 compression of untrained images with a PSNR = 26.9 dB is presented in by Marsi (1991). 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 by Sonehara (1989). Huang (1991) compares backpropagation networks with recirculation networks and the DCT. 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. Feng (1991) presents a VLSI implementation of a neurovector quantization/codebook algorithm. Kohno (1990) suggests the use of a nonlinear mapping function with parameters that are learned in order to achieve better image compression in a standard backpropagation network. Namphol (1991) uses a backpropagation-based nested training algorithm to compress. 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 (PSNR = 22.89 dB), 64:1 (PSNR = 15.15 dB) to 256:1 (PSNR = 10.44 dB). For previously "unseen" images, results are given with the following ratios and qualities: 8:1 (PSNR = 18.13 dB) to 64:1 (PSNR = 12.93 dB). 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) (Gelenbe and Sungur 1984) where we obtain a PSNR close to 30 db for a 16:1 ratio.

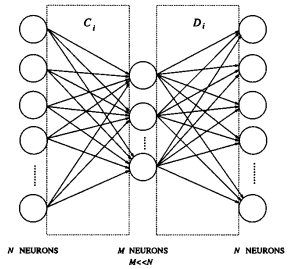
Motion detection and compensation are key issues when one deals with moving images. Motion compensation provides for a great deal of the compression in the MPEG standard. By using motion compensation, MPEG can code the blocks in a frame in terms of motion vectors for the blocks in the previous and/or next frames. To compensate for motion, the motion must be estimated by block matching over the area local to the block under consideration. Exhaustive searches that 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 motion cannot be fully compensated in real time since the future frame must 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 the motion compensation problem in two dimensions has been examined. Courellis (1990) presents a neural network for motion detection. However, it only works for a 1D case, and the author states that problems arise when the approach is extended to 2D 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. Chiang (1990) presents a neural network method for motion estimation. 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 because the 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 (Chiang 1990,) 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



3

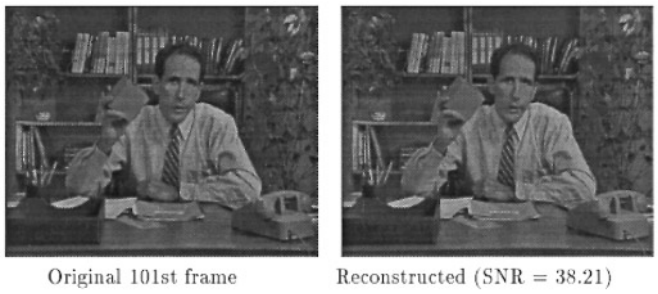


4

Fig. 3. An arbitrarily large image compressed with a neural encoder/decoder

Fig. 4. Compression/decompression pair in a neural network

5



6

Fig. 5. Still image compressed at a 16:1 ratio(0.5 bits/pixel) with a random neural network

Fig. 6. Original and reconstructed 101st frames in the Salesman sequence. In the motion detection scheme, $d = 1$

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 that Gelenbe and Sungur (1994) describe, the input, intermediate, and output images are subdivided into equally-sized blocks, and they are compressed block by block (Fig. 3), as in JPEG and MPEG. This has the desirable effect of reducing the learning time of the neural network. 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 random neural network with a feed-forward encoder/decoder and with one intermediate layer as shown in Fig. 4. 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 linearly map the $[0, 255]$ interval into the $[0, 1]$ interval since the gray level of each image pixel is transformed into a real-valued excitation level of a neuron (and vice versa). This is done by simply dividing the pixel value by 255.0. 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 with the algorithm described by Gelenbe (1993) and with 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 PSNR=23 dB to more than 30 dB for 16:1 compression (0.5 bits/pixel). As an example, Fig. 5 shows our results with 16:1 compression for the 512×512 8-bit *Peppers* image (Gelenbe and Sungur 1984).

2.1 Motion detection

We deal with sequences of image frames representing a moving video sequence. Thus, very often, a substantial part of an image, such as the background, basically does not move – except for noise that may originate at various levels, including the imaging devices. However, 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 to detect motion.

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 perceptible to the human eye is approximately $d = 1$. d Also affects the compression ratio by determining how many blocks in a frame

will be sent. 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 contrast, small blocks incur a larger overhead for transmission, but improve frame quality. Blocks of size 8×8 were chosen as a good compromise.

For the data we present, 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 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 2. 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 in 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 6 shows the original and the reconstructed 101st frame of the sequence with d = 1. In Fig. 7a, the PSNR is plotted as a function of frame number for d = 1. Similarly, Fig. 7b 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 is combined with the actual neural compression of frames to achieve large compression ratios and satisfactory image quality.

2.2 Compression for moving images

We now describe and evaluate the complete compression scheme for video sequences of natural images. We use a combination of the motion detection scheme described earlier and our adaptive still block-by-block (Fig. 3) technique that includes compression/decompression with a random neural network. Specifically, our compression scheme uses three networks:

- The first network scans successive blocks in sequence, and identifies those blocks where motion has taken place, as already described. If a block is considered identical to the same box in the previous frame, it is not compressed or transmitted.
- The second network compresses the blocks that have been identified by the first network. In fact, the second network is a set of distinct neural compression networks $C_1, \ \ldots, C_L$. Each network has been designed and trained to compress blocks at a different compression level. Each of these networks compresses the box in parallel. The compression level is selected 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 that yields the largest compression ratio at a quality level of Q or better, chosen to be acceptable for the particular application, is selected and the compressed block is transmitted. For gray-level images, Q is formulated as a SNR value. Figure 8 shows the block diagram of the adaptive, still image, compression network.

Note that, with the exception of the learning phase for training all of these networks with an unseen image (in our case the well-known *Lena* image), all the operations that have been outlined are carried out "on-the-fly", that is in real time as each block leaves the sender, and as each compressed block goes into the receiver and is decompressed (Fig. 1). The relationship between any two compression/decompression networks C_i, D_i is shown in Fig. 4.

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 train the network C_i further in on-line mode. In this case, D_i's weights are *not* changed, and only C_i's weights are updated.

At the "receiving or decompression" end, if the transmitter has indicated that the current block is identical to the same block in the previous frame, then the previous frame's block is placed in the corresponding position of the output image. Otherwise, the compressed block is received. Implicitly (through the block's size) or explicitly (via a variable i that accompanies the block) the compression level used is known to the receiver. Therefore, the appropriate network D_i is used to decompress the block, which is then placed in sequence in the output image.

3 Experimental results

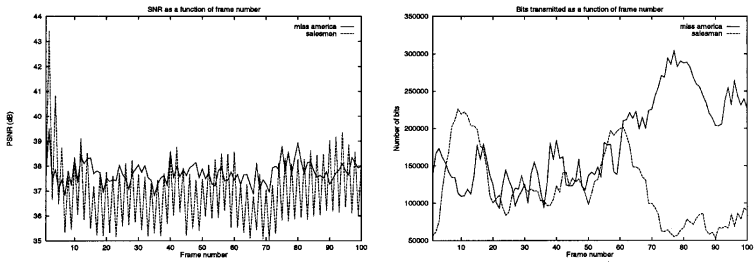
The experimental results we now present were obtained with three compression/decompression pairs (L = 4) with compression ratios of 4:1, 8:1, 16:1 and 32:1. The target quality for the decompressed image is set to the SNR value of Q = 30. The gray-level video sequences used in the tests are *Miss America* and *Salesman*, mentioned previously.

In Table 2 we summarize the experimental results for various 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.

It can be seen in Table 2 and Figs. 9a and b that both the compression ratio and the video quality are almost linear functions of d. This linearity will allow future implementations to adaptively control the trade-off between video quality and compression ratio in an informed manner. Current

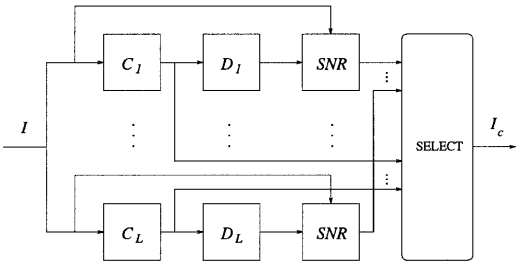
Table 1. Compression ratios obtained *only* by motion detection as a function of the sensitivity of the motion detector d

d	Miss America				Salesman			
	Compression ratio		Frame SNR		Compression ratio		Frame SNR	
	Total	Steady state	Minimum	Average	Total	Steady state	Minimum	Average
0.5	2.25	2.28	38.78	39.80	3.01	3.07	37.38	39.16
1.0	4.44	4.59	36.82	37.72	6.55	6.94	35.04	36.97
1.5	6.06	6.38	35.72	36.67	9.23	10.06	33.66	35.70
2.0	7.25	7.74	34.57	35.84	11.26	12.55	32.77	34.73
2.5	8.42	9.10	33.91	35.23	13.08	14.88	31.99	33.81
3.0	9.53	10.41	33.63	34.62	14.70	17.04	31.41	33.25
3.5	10.60	11.73	33.02	34.19	16.32	19.29	30.84	32.71
4.0	11.71	13.11	32.69	33.74	18.01	21.71	30.60	32.25
4.5	12.82	14.54	32.37	33.29	19.75	24.30	30.05	31.80
5.0	13.96	16.04	32.08	32.98	21.38	26.86	29.77	31.39

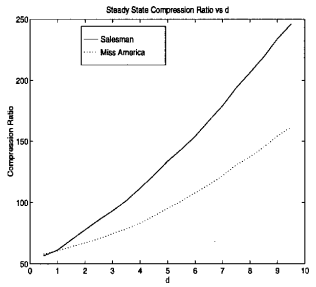


7a

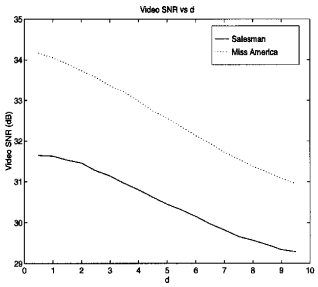
7b



8



9a



9b

Original 101st frame

Reconstructed (SNR = 33.82)

10

Fig. 7a,b. Experimental results for motion detection with d = 1: **a** PSNR as a function of the frame number; **b** number of bits transmitted as a function of the frame number

Fig. 8. Block diagram of the adaptive, still image, compression network

Fig. 9. Steady state compression ratio as a function of d; **b** Average video pek signal-to-noise ratio (PSNR) as a function of d

Fig. 10. Original and reconstructed 101st frames in the Miss America sequence with d = 1.5 and Q = 30

video compression techniques are not linear in terms of their compression parameters. This means that, while one could adaptively control either the compression ratio or the video quality in MPEG and H.261, the effects on the compression ratio of adjusting the quality of the video sequence could not be as easily predicted as in the adaptive neural video compression (ANVC) scheme.

In Fig. 10 we show the original and the reconstructed 101st frame of *Miss America* using the complete scheme described with d = 1.5 and Q = 30. Figure 11 indicates the variation of the compression ratio over time. Figure 12 shows the running average compression ratios and the running average bits per pixel for a run length of 1000, based on the *Miss America* sequence with d = 2 and Q = 30. In Fig. 13a, the SNR is plotted as a function of frame number for d = 2, Q = 30. Figure 13b 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

Table 2. Compression ratios obtained by motion detection and compression with the image-quality level Q = 30 as a function of threshold d

d	Miss America				Salesman			
	Compression ratio		Frame SNR		Compression ratio		Frame SNR	
	Total	Steady state	Minimum	Average	Total	Steady state	Minimum	Average
0.5	37.06	57.96	33.62	34.17	36.51	56.62	30.56	31.65
1.0	38.16	60.72	33.51	34.05	38.37	61.27	30.50	31.63
1.5	39.39	63.93	33.29	33.90	41.46	69.64	30.37	31.53
2.0	40.55	67.08	33.05	33.73	44.19	77.78	30.26	31.46
2.5	41.76	70.50	32.80	33.57	46.65	85.82	30.05	31.27
3.0	43.13	74.54	32.74	33.36	48.73	93.23	29.81	31.14
3.5	44.44	78.59	32.53	33.21	50.87	101.48	29.64	30.96
4.0	45.84	83.11	32.29	32.98	53.25	111.54	29.54	30.80
4.5	47.56	88.99	32.13	32.74	55.53	122.10	29.32	30.62
5.0	49.25	95.18	31.96	32.56	57.78	133.68	29.18	30.45
5.5	50.75	100.98	31.65	32.35	59.50	143.38	29.06	30.31
6.0	52.39	107.79	31.28	32.13	61.23	153.94	28.86	30.15
6.5	53.86	114.24	31.10	31.94	63.15	166.81	28.70	29.97
7.0	55.46	121.79	30.95	31.72	64.81	179.06	28.51	29.82
7.5	57.20	130.62	30.60	31.55	66.63	193.88	28.39	29.66
8.0	58.42	137.17	30.50	31.38	68.02	206.29	28.33	29.57
8.5	59.79	145.07	30.26	31.23	69.35	219.16	28.17	29.46
9.0	61.28	154.28	30.07	31.08	70.75	233.90	28.08	29.34
9.5	62.38	161.55	29.98	30.95	71.81	246.03	28.07	29.29

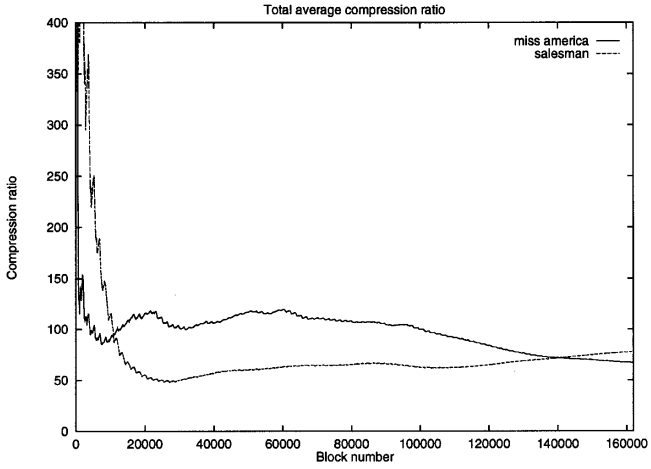


Fig. 11. Total average compression ratio as a function of the block number with d = 2 and Q = 30

sequence 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*).

3.1 Simulating ATM traffic from the compressed video sources

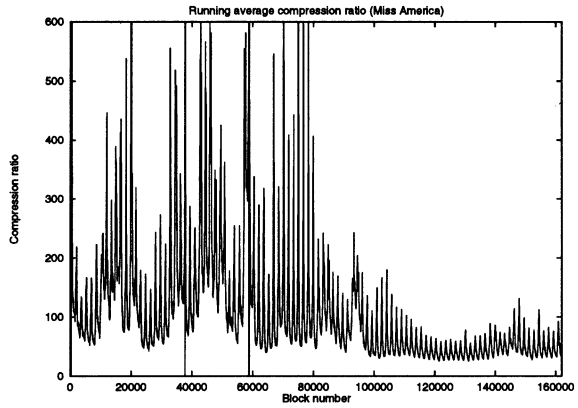
We now turn to a study of the traffic being generated by our method. Since the video sequences produce images at a rate of 30 frames/s, and because each transmitted frame is composed either of compressed blocks or of positional information concerning a block that does not need to be transmitted, it is 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 48-byte payload of ATM cells in such a way that any one block cannot span two different cells.

Thus, on Fig. 14 we show the traffic rate in ATM cells/s that is generated for the *Miss America* video sequence in a few seconds. We observe the highly unpredictable nature of the traffic and its time-varying behavior. Figure 15 presents the corresponding autocorrelation function of cell traffic generated by our compression method for d = 0.5 and d = 2, indicating a linear decrease in a few seconds. These results are presented for the same target image quality Q = 30.

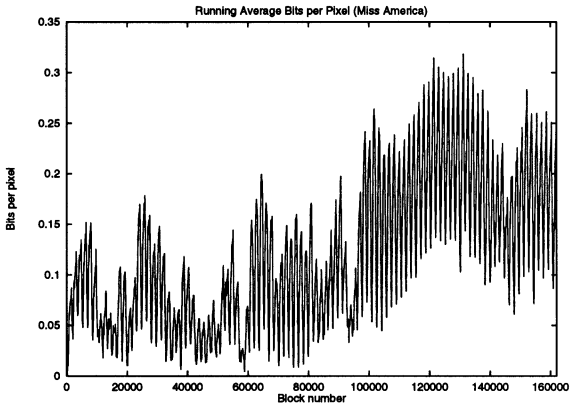
Much ATM traffic can be expected to travel over relatively slow legacy networks for a number of years to come, especially for portions of the network that are geographically close to the user generating the video traffic. Therefore, we also examine the behavior of a finite ATM buffer queue for these traffic streams and measure the cell loss rate for various values of d, different buffer sizes, and various speeds at which the buffer is being emptied. These results are summarized on Figs. 16 and 17 for the *Miss America* sequence. We have purposely chosen a slow legacy network speed of 64 kBytes/s to evaluate the losses observed on a link when a link capacity that 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 buffer sizes are small. In order to see what the effect of doubling the link speed can be, on Fig. 18 we plot the ATM cell loss rate for a 128 kByte/s with the same video sequence and with d = 3. Clearly, by comparing results with Fig. 18, the cell loss can be substantially reduced by doubling the link speeds, even when they are relatively small.

4 Conclusions

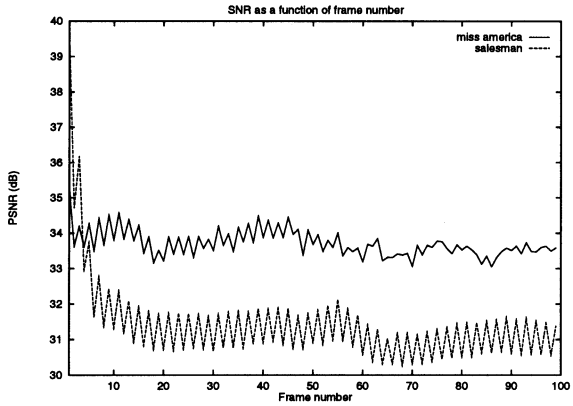
In this paper we have described an adaptive neural technique for video compression and have studied its traffic characteristics. Our compression method is designed to meet quality requirements with respect to the decompressed image



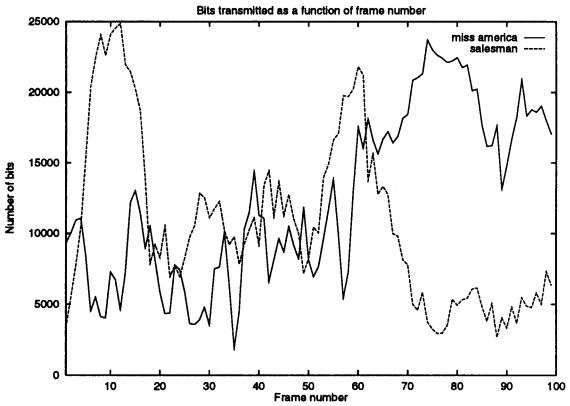
12a



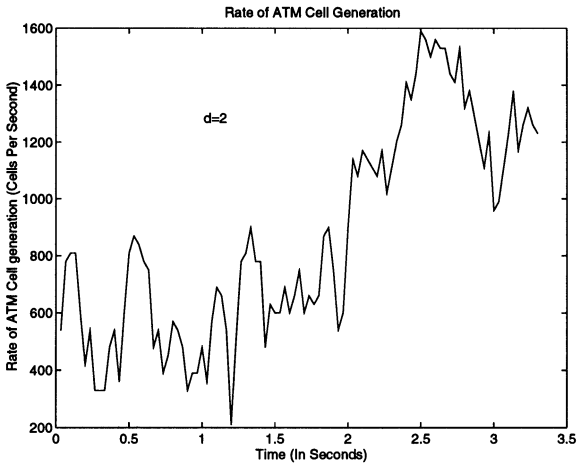
12b



13a



13b



14

Fig. 12a,b. Experimental results for the Miss America sequence with $d = 2$ and $Q = 30$: **a** running average compression ratio as a function of the block number; **b** running average bits per pixel as a function of block number

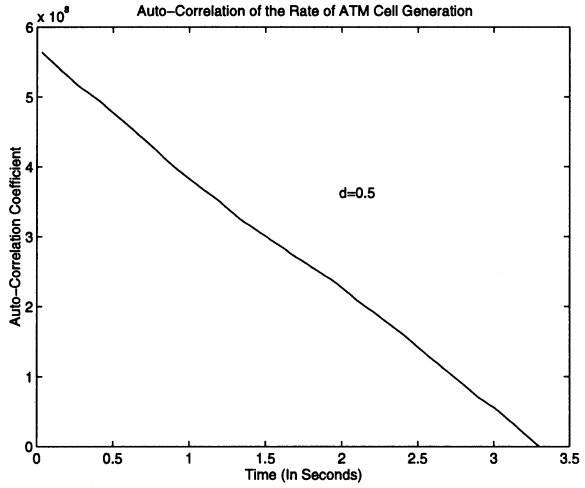
Fig. 13a,b. Experimental results with $d = 2$ and $Q = 30$: **a** peak signal-to-noise ratio (PSNR) as a function of frame number; **b** number of bits transmitted as a function of frame number

Fig. 14. Asynchronous transfer mode (ATM) traffic rate with $d = 2$

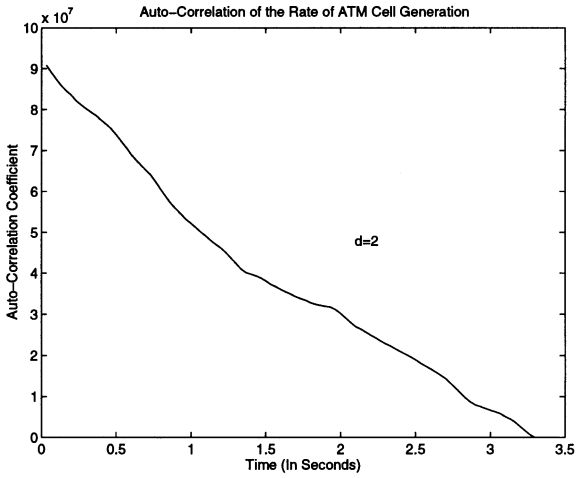
at the receiver. It can vary the traffic rate it generates into the network by modifying a simple parameter that is used to detect motion in successive frames. It can also do this by modifying the image quality requirement. This method is computationally inexpensive in that motion detection is used in place of the motion estimation used by methods such as H.261 and MPEG. Furthermore, each block is compressed in $O(n^2)$ time. Timed results indicate that this method is several times faster than the H.261 and MPEG compression schemes. Figure 19 provides an experimental comparison of the time required to compress each frame in ANVC and H.261 for practical video sequences and shows that ANVC is more than six times faster.

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

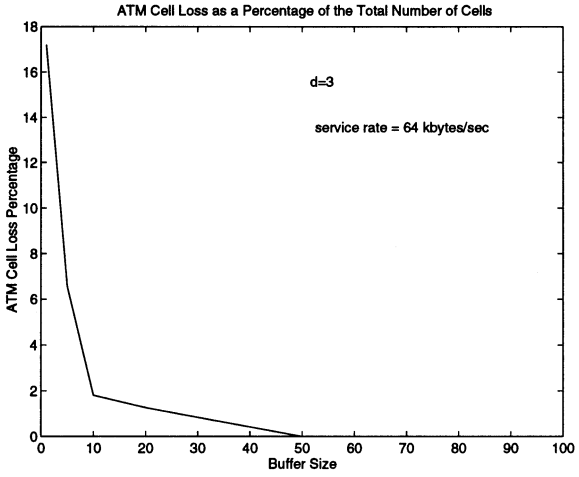
We also present experimental data concerning the ATM traffic that our method will generate, including measurement of cell traffic rates, autocorrelation of the traffic and buffer



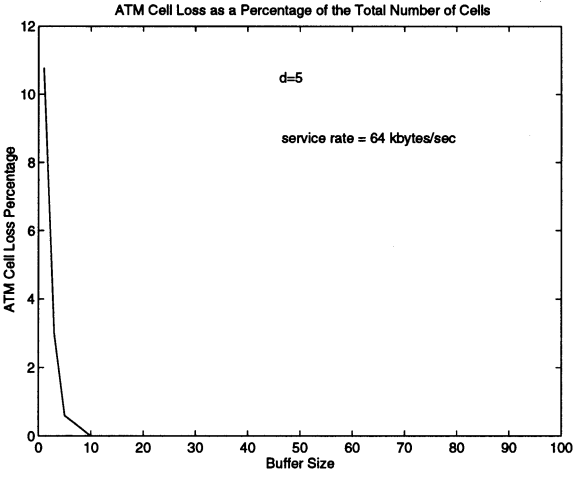
15a



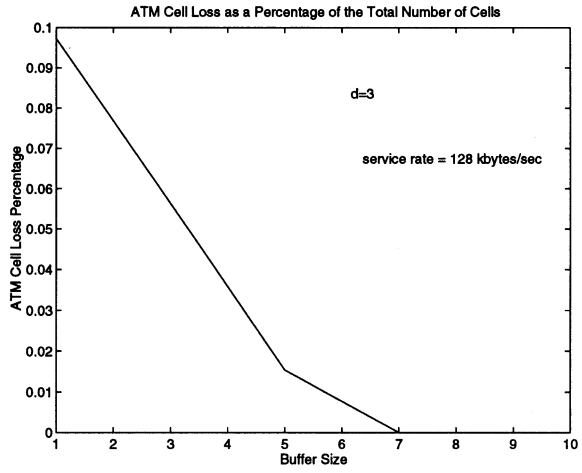
15b



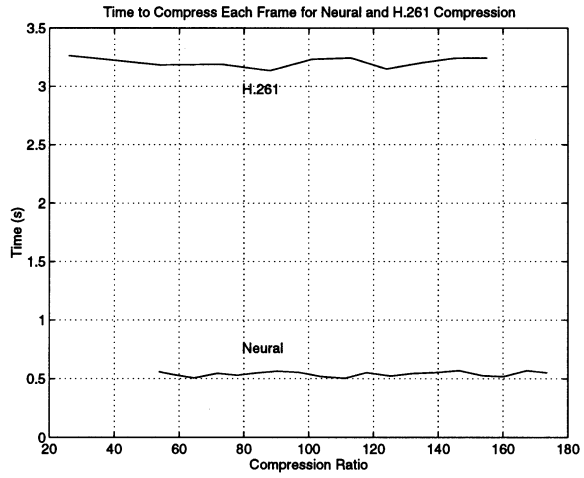
16



17



18



19

Fig. 15a,b. Autocorrelation function of ATM cell traffic with **a** d = 0.5 and **b** d = 2

Fig. 16. Percentage of lost cells vs. buffer size for a 64 kByte/s link with d = 3

Fig. 17. Percentage of lost cells vs. buffer size for a 64 kByte/s link with d = 5

Fig. 18. Percentage of lost cells vs. buffer size for a 128 kByte/s link with d = 3

Fig. 19. Time to compress each frame in ANVC and H.261 compression schemes as a function of compression ratio

overflow for ATM buffers of finite sizes. These results are based on real-image sequences that are used to drive our moving image compression method.

Many further improvements of the basic method investigated in this paper can be envisioned, 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 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, 1990) 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 reaches 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 an excitation or an inhibition spike, loses one unit of potential, going from a state of value 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 denote by k and k_i arbitrary values of the state vector and of the ith neuron's state.

Neuron i will "fire" (i.e., become excited and send out spikes) if its potential is *positive*. The spikes are sent out at a rate $r(i)$, with independent, identically and exponentially distributed interspike intervals. Spikes 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 a 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 at neuron i at rates $\Lambda(i)$ and $\lambda(i)$, respectively.

This is a "recurrent network" model; that is, it 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 behavior 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, sends spikes to neuron i at a frequency $w_{ij} = w_{ij}^+ + w_{ij}^-$. These spikes are emitted at exponentially distributed random intervals. In turn, each neuron behaves as a nonlinear *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:

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

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 $i = 1, \ldots, n$ satisfy the system of nonlinear simultaneous equations:

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

Let $k(t)$ be the vector of neuron potentials at time t and $k = (k_1, \ldots, k_n)$ 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 non-negative solution $\{\lambda^+(i), \lambda^-(i)\}$ exists for Eqs. 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 that are most useful for computational purposes, that is, the probabilities that each neuron is excited, are directly obtained from:

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

Let us now describe the learning algorithm we use in this study. It is based on the algorithm described in (Gelenbe 1993).

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 $\boldsymbol{\iota} = \{\iota_1, \ldots, \iota_K\}$, and $\iota_k = (\Lambda_k, \lambda_k)$ are pairs of positive and negative signal flow rates entering each neuron:

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

The successive desired outputs are the vectors $\mathbf{Y} = \{y_1, \ldots, y_K\}$, where each vector $y_k = (y_{1k}, \ldots, y_{nk})$, 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 a neuron j from network output, and hence from the error function, it suffices to set $a_j = 0$.

Both of the $n \times n$ 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 a new value $\mathbf{W}_k^+$ and $\mathbf{W}_k^-$ of the weight matrices for each input $\iota_k = (\Lambda_k, \lambda_k)$. The computation uses 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 are updated as follows:

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

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

1. q_{ik} Is calculated with the input ι_k and $w(u,v) = w_{k-1}(u,v)$, in Eq. 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 Eq. 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 $\mathbf{q} = (q_1, \ldots, q_n)$, and define the $n \times n$ matrix

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

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

$$\begin{aligned} \boldsymbol{\gamma}^+(u,v) &= [\gamma_1^+(u,v), \ldots, \gamma_n^+(u,v)] \ , \\ \boldsymbol{\gamma}^-(u,v) &= [\gamma_1^-(u,v), \ldots, \gamma_n^-(u,v)] \end{aligned}$$

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

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

where $\mathbf{I}$ denotes the $n \times n$ 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 an 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 Eqs. 3 with these values.
3. Solve the system of linear Eqs. 8 with the results of step 2.
4. Using Eq. 7 and the results of steps 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 *non-negativity* 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 step k; in the next stop $k+1$ we 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.

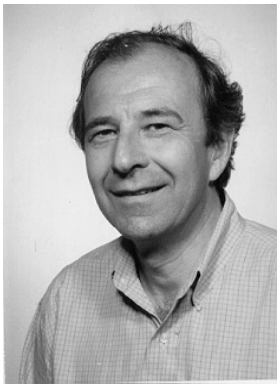
This general scheme can be specialized to feed-forward networks yielding a computational complexity of $O(n^2)$, rather than $O(n^3)$, for each gradient iteration.

References

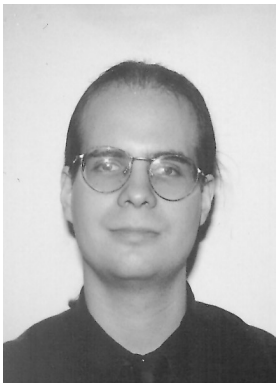
1. Adelson EH, Simoncelli E (1987) Orthogonal pyramid transforms for image coding. Visual Commun Image Processing II, Proceedings of SPIE 845: 50–58
2. Bolot J-C, Turletti T (1993) A rate control mechanism for packet video in the Internet. Proceedings of IEEE INFOCOM'94, Toronto, 3: 1216–1223
3. Carrato S (1992) Neural networks for image compression. In: Gelenbe E (ed) Neural networks: advances and applications 2. Elsevier North-Holland, pp 177–198
4. Chen C-T, Wong A (1993) A self-governing rate buffer control strategy for pseudoconstant bit rate video encoding. IEEE Trans Image Processing 2: 50–59
5. Chiang YW, Sullivan BJ (1990) Motion estimation using a neural network. Proceedings of the IEEE International Symposium on Circuits and Systems, New Orleans, IEEE Piscataway, NJ, pp 2516–2519
6. Cottrell GW, Munro P, Zipser D (1989) Image compression by back-propagation: an example of extensional programming. In: Sharkey NE (ed) Models of cognition: a review of cognition science. Ablex: Norwood, NJ
7. Courellis SH, Marmarelis VZ (1990) An artificial neural network for motion detection and speed estimation. Proceedings of the International Joint Conference on Neural Networks (IJCNN'90), San Diego, CA, IEEE Piscataway, NJ, pp 407–421

8. Daugman JG (1988) Relaxation neural network for non-orthogonal image transformations. Proceedings of the International Conference on Neural Networks, San Diego, CA, IEEE Piscataway, NJ, pp 547–560
9. Fendick K, Rodriguez M, Weiss A (1992) Analysis of a rate based control strategy with delayed feedback. Proceedings of ACM SIGCOMM'92, Baltimore, ACM, New York, pp 136–142
10. Feng WC, Shen BJ, Chen OT(1991) Real-time neuroprocessor for adaptive image compression based upon frequency-sensitive competitive learning. Proceedings of the International Joint Conference on Neural Networks (IJCNN'91), Seattle, WA, IEEE Piscataway, NJ, pp 429–435
11. Gilge M, Gusella R (1991) Motion video coding for packet-switching networks. Proceedings of the SPIE Conference on Visual Communications and Image Processing, Boston, Int Soc for Optical Eng, Bellingham, pp 592–603
12. Jeffay K, Stone DL, Talley T, Smith D (1992) Adaptive best effort delivery of audio and video data across packet switched network. Proceedings of 3rd International Workshop on Network and OS Support for Digital Audio and Video, San Diego, Springer Verlag, New York, pp 3–14
13. Kanakia H, Mishra P, Reibman A (1989) An adaptive congestion control scheme for real-time video transport. Proceedings of ACM SIGCOMM'93, San Francisco, ACM, New York, pp 20–31
14. Gelenbe E (1989) Random neural networks with negative and positive signals and product form solution. Neural Comput 1: 502–511
15. Gelenbe E (1990) Stability of the random neural network model, Neural Comput 2: 239–247
16. Gelenbe E (1993) Learning in the recurrent random neural network Neural Comput 5: 154–164
17. Gelenbe E, Sungur M (1994) Random Network Learning and Image compression. Proceedings – IEEE Int Conf on Neural Networks, Orlando, IEEE Piscataway, NJ, pp 3996–3999
18. Gray RM (1984) Vector quantization. IEEE ASSP Magazine 1: 4–29
19. Huang SJ, Koh SN, Tang HK (1991) Image data compression and generalization capabilities of backpropagation and recirculation networks. Proceedings of the International Symposium on Circuits and Systems, Singapore, IEEE Piscataway, NJ, pp 1613–1616
20. Jacquin AE (1992) Image coding based on a fractal theory of iterated contractive image transformations. 1: 18–30
21. Kohno R, Arai M, Imai I (1990) Image compression using a neural network with learning capability of variable function of the neural unit. SPIE Visual Commun Image Processing, SPIE, Bellingham, pp 69–75
22. Kohonen T (1987) Self organization and associative memory. Springer, Berlin
23. Kunt M, Benard M, Leonardi R (1987) Recent results in high-compression image coding. IEEE Trans Circuits Syst 34: 1306–1336
24. LeGall D (1991) MPEG: A video compression standard for multimedia applications. Commun ACM 34: 46–58
25. Marsi S (1991) Improved neural structures for image compression. Proceedings of the International Conference on Acoustic Speech and Signal Processing (ICASSP'91), Toronto, IEEE Piscataway, NJ, pp 2821–2824
26. Naillon M (1989) Advances in Neural Processing Systems. Morgan-Kaufmann, San Mateo
27. Namphol A (1991) Higher-order data compression with neural networks. Proceedings of the International Joint Conference on Neural Networks (IJCNN'91), Seattle, IEEE Piscataway, NJ, pp 55–59
28. Nasrabadi NM, Feng Y (1988) Vector quantization of images based upon Kohonen self organizing feature maps. Proceedings of the International Conference on Neural Networks, San Diego, IEEE Piscataway, NJ, pp 101–108
29. Ramamurthi B, Gersho A (1986) Classified vector quantization of images. IEEE Trans Commun 34: 1105–1115
30. Rumelhart DE, McClelland JL, PDP Research Group (1986) Parallel Distributed Processing. vol 1, 2. MIT Press, Cambridge, Mass
31. Sonehara N (1989) Image data compression using a neural network model. Proceedings of the International Joint Conference on Neural Networks (IJCNN'89), Washington, DC, IEEE Piscataway, NJ, pp 35–41
32. Turletti T (1993) H. 261 software coder for videoconferencing over the Internet. INRIA Research Report 1834, Rocquencourt
33. Wakeman L (1992) A combined admission and congestion control scheme for variable bit-rate video. University College London (UCL) Technical Report RN/92/93, University College London, London
34. Wakeman L (1993) Packetized video: options for interaction between the user, the network and the codec. Comput J 36:55–67
35. Wallace GK (1991) The JPEG still picture compression standard. Commun ACM 34: 30–44
36. Woods J, O'Neil SD (1986) Subband coding of images. IEEE Trans Acoustics Speech Signal Processing, 34: 1278–1288
37. Zettler W, Huffman J, Linden DCP (1990) Application of compactly supported wavelets to image compression. Image Processing Algorithms and Techniques, Proceedings of SPIE 1244: 150–160

EROL GELENBE holds a Bachelor's degree in Electrical Engineering with High Honors from the Middle East Technical University in Ankara, Turkey, a Master's and PhD in Electrical Engineering from Polytechnic University in New York, and a DSc from the University of Paris in Computer Science. At Duke University he is the Nello L. Teer Jr. Professor of Electrical and Computer Engineering, and also a Professor of Computer Science. His interests include computer-communication networks, parallel and distributed computer systems, computer performance analysis, artificial neural networks, and image processing. Dr. Gelenbe is an Associate Editor of *Acta Informatica, Telecommunication Systems, Performance Evaluation, Journal de Recherche Opérationelle, Information Sciences,* and *Simulation Practice and Theory*. He is a Fellow of the IEEE.



MERT SUNGUR received his BS and MS degrees in Electrical and Electronics Engineering from the Middle East Technical University, Ankara, Turkey, in 1990 and 1992, respectively. He has worked as a visiting scholar at Duke University for two semesters. Currently he is pursuing the PhD degree in the Department of Electrical and Electronics Engineering, Middle East Technical University. His research interests include artificial neural networks, computer vision, and image processing. He is a student member of IEEE, IEEE Computer Society, and IEEE Control Systems Society. He has been the secretary of IEEE Computer Society, Turkey Chapter, since 1991.



CHRISTOPHER CRAMER is a doctoral candidate in Electrical and Computer Engineering at Duke University, Durham, N.C. He holds a BSEE degree from Louisiana State University, Baton Rouge, La., and an MS degree in Electrical Engineering from Duke University. His interests include neural networks, image compression, and very low bit-rate video compression.

PAMIR EMRE GELENBE was born in Paris in 1973. He received his BS degree in Electrical Engineering, Summa Cum Laude, with Distinction in Electrical Engineering, in 1995. At Duke University he founded *The Engineering Dispatch*, a quarterly student magazine, and was President of the Turkish Students Society. He is fluent in Turkish and German, as well as in English and French. His technical interests are in telecommunication and computer-communication networks. In the summer of 1995 he interned with the consulting company Deloite & Touche in Beirut, Lebanon, as part of a World Bank funded program to redesign the Lebanese telecommunication networks. He is a student at the Ecole Centrale in Paris working towards his French Engineer's degree. Pamir Gelenbe is a Student Member of the IEEE.