# Video Quality and Traffic QoS in Learning Based Subsampled and Receiver-Interpolated Video Sequences *

Christopher E. Cramer
Electrical and Computer
Engineering Department
Duke University
Durham, NC 27708-0291
cec@ee.duke.edu

Erol Gelenbe, *Fellow IEEE*
School of Computer Science
University of Central Florida
Orlando, FL 32816
erol@cs.ucf.edu

September 10, 1999

**Abstract**

Sources of real-time traffic are generally highly unpredictable with respect to the instantaneous and average load which they create. Yet such sources will provide a significant portion of traffic in future networks, and will significantly affect the overall performance of and Quality-of-Service. Clearly high levels of compression are desireable as long as video quality remains satisfactory, and our research addresses this key issue with a novel learning based approach. We propose the use of neural networks as post-processors for any existing video compression scheme. The approach is to interpolate video sequences and compensate for frames which may have been lost or deliberately dropped. We show that deliberately dropping frames will significantly reduce the amount of offered traffic in the network, and hence the cell loss probability and network congestion, while the neural network post-processor will preserve most of the desired video quality. Dropping frames at the sender or in the network is also a fast way to react to network overload and reduce congestion. Our interpolation techniques at the receiver, including neural network based algorithms, provide output frame rates which are identical to (or possibly higher than) the original video sequence's frame rate. The resulting video quality is essentially equivalent to the sequence without frame drops, despite the loss of a significant fraction of the frames. Experimental evaluation using real video sequences is provided for interpolation with a connexionist neural network using the back-propagation learning algorithm, the Random Neural Network (RNN) [21] in a feed-forward configuration with its associated learning algorithm, and cubic spline interpolation. The experiments show that when more frames are being dropped or lost, the RNN performs generally better than the other techniques in terms of resulting video quality and overall performance. When the fraction of dropped frames is small, cubic splines offer better performance. Experimental data shows that this receiver-reconstructed subsampling technique significantly reduces the cell loss rates in an ATM switch for different buffer sizes and service rates.

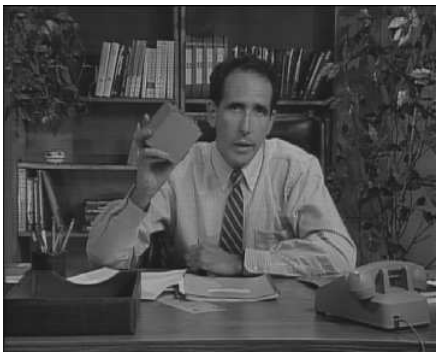
**Keywords:** Neural Networks, Video Compression, Subsampling, Receiver-Based Interpolation

*This work was partially supported by the US Office of Naval Research under grant number N00014-97-1-0112.

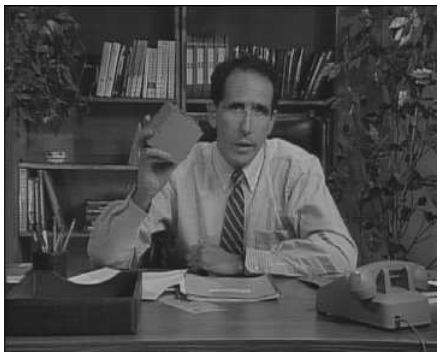
# 1 Introduction

Video, which can originate in distance learning applications, video-conferencing, video-surveillance, entertainment, and in a variety of other applications, is becoming a major source of real-time network traffic. Modern video encoding techniques generate variable bit rates, since they take advantage of different rates of motion in scenes in addition to using lossy compression within individual frames.

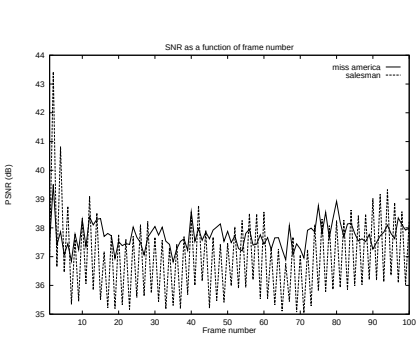
To illustrate these effects, in Figure 1 we show an original and reconstructed (after compression and decompression) frame in the well-known SALESMAN sequence using simple motion detection based compression, and the resulting time varying Video Quality (expressed as the instantaneous frame by frame Signal-to-Noise-Ratio) and Bit-Rate for the SALESMAN and MISS AMERICA video sequences. The third figure on the bottom row shows the resulting ATM Cell Rate for the compressed SALESMAN Sequence. Variations in measurements for subsequent frames and over time, as well between the two different video sequences at any given time instant, can be very significant [47] and are in general unpredictable for different video sequences.



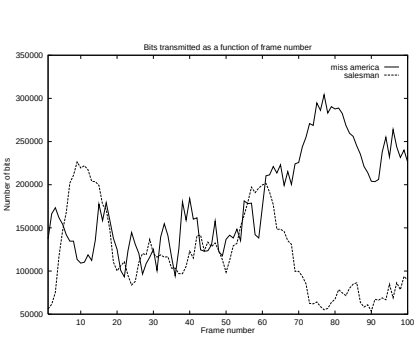
Original 101st frame



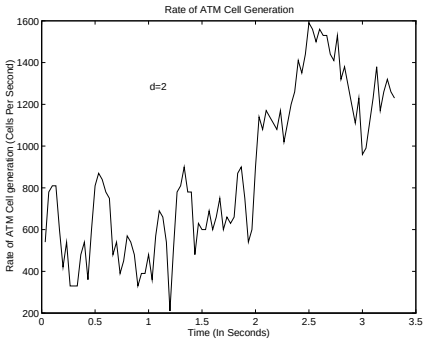
Reconstructed 101st frame (SNR = 38.21)




PSNR vs Frame Number




Bits Transmitted vs Frame number




ATM Cell Rate vs Time

Figure 1: Above: Original and reconstructed 101st frames in the SALESMAN sequence using simple motion detection based compression, Below: Resulting Time Varying Video Quality and Bit-Rate for the SALESMAN and MISS AMERICA, and ATM Cell Rate for the SALESMAN Sequence. Variations in measurements for subsequent frames and over time, as well between the two different video sequences at any given time instant, can be very significant [47].

To further illustrate the time variations to be expected in video traffic, on Figure 2 we show the traffic rate in ATM cells/second which is being generated for the *Miss America* video sequence

over a few seconds, where the sequence has been compressed using an adaptive neural network technique (Adaptive Neural Video Compression or ANVC) which we reported in an earlier paper [47]. We again observe the highly variable nature of the traffic and it's time-varying behaviour. Figure 2 also presents the corresponding autocorrelation function of ATM cell traffic.

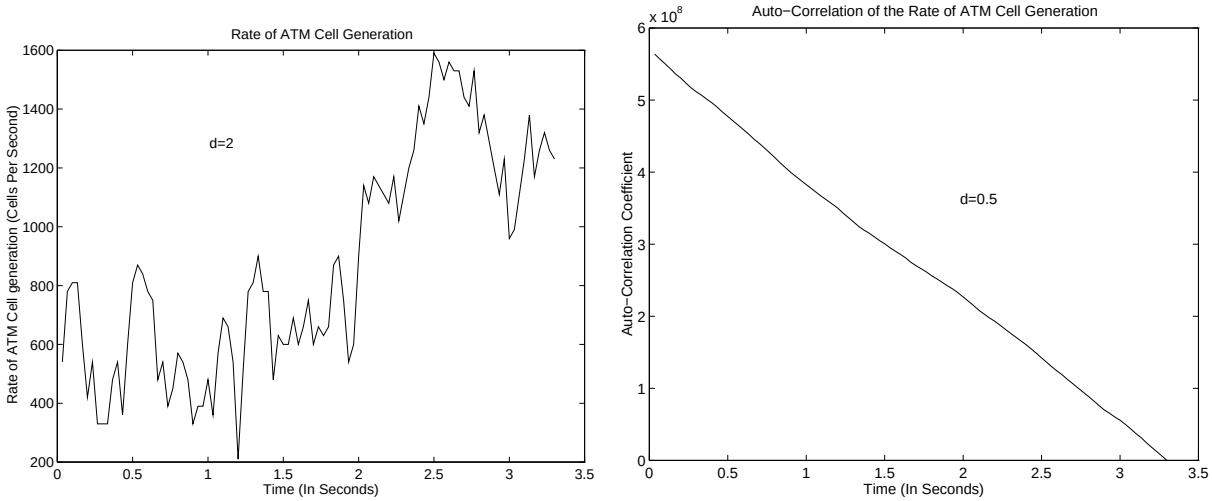



Figure 2: ATM traffic rate (Left) and autocorrelation function of ATM cell traffic rate (Right) for Adaptive Neural Video Compression (ANVC) from [47]

Video will typically traverse a variety of networks from source to destination. This may include ad-hoc networks set up inside a user's private domain with diverse technologies such as wireless, coaxial and LAN, then public wireless, ATM, or telephone networks, as well as the Internet. The users of video will be dealing with a variety of networks which are unknown to them, and through which they will have to transmit and receive video whose quality must remain satisfactory to the end user. Thus end-to-end quality of service (QoS) of video is highly sensitive to bit rates which are available, and also to other network performance metrics such as end-to-end delays, cell loss rates in ATM, communication hand-off and blocking problems in wireless, as well as second order performance characteristics such as jitter. Video compression standards use motion prediction and compensation to encode the sequence of frames. Such techniques create additional performance constraints, since they generate encoded frames of varying types and sizes (as in MPEG), and place a major burden on the correct reception of certain critical frames. Despite the fact that successive frames may arrive at random and at irregular intervals, that delays between successive frame arrivals may vary widely and be out of synchrony with the regular time intervals needed for good viewing quality, that some frames may arrive out of order due to adaptive routing or multipath effects, and that some frames may be completely or partially lost or corrupted, ideally one would like to offer viewing which gives the end user the illusion of a smooth arrival rate of uncorrupted frames.

In this paper we propose and demonstrate novel adaptive techniques which are designed to offer acceptable end-to-end video quality for the user, despite the "drop" (subsampling) of a significant fraction of the video frames. These techniques can be used when video frames are accidentally lost. They can also be used when frame dropping is carried out deliberately to increase the compression ratio. Our approach is designed for the case where the majority of frames is "dropped", so that only one in $S$ frames is actually transmitted or received. Such techniques can serve as pre- and post-processors for other transmission techniques, whether they transmit raw video, use some lossless

compression method, or whether they make use of some lossy video compression standard.

Our approach considers the consecutive values of any pixel $\pi(x, y)$ in a video sequence as a function $f(x, y, t)$ of time $t$. Here time is continuous in the abstract, since the video sequence represents some "physical (or virtual) reality", and the value of that pixel in the $i-th$ (uncompressed) video frame is the sample value $f(x, y, iT)$, where $T = F^{-1}$ and $F$ is the frame rate. Suppose that $f$ had the property that its values for all $t \in [a, b]$ can be computed by knowing $f$ at only a few points in $[a, b]$ – then this would provide us with a very powerful tool for reconstructing the video sequence from just a few reliable frames. Of course, this idea would be limited by the sampling theorem; however the type of motion we are considering (e.g., people moving around in a scene, or ground vehicles and people observed in motion from a distance) is slow enough to be amenable to this approach [1].

Some first steps in this direction were taken in our recent research [47] where we have used function approximation using linear interpolation as a way to reconstruct video from temporally subsampled frame sequences. In this paper we show how learning neural networks can provide improved video sequences. The measure of improvement is a higher compression ratio for the same reconstructed video quality, or a higher video quality for a given compression ratio.

Specifically, we will develop novel neural network based function approximation techniques which will construct an approximation $g(x, y, t)$ to the function $f(x, y, t)$ in an interval $t \in [a, b]$, using only a "small" finite number of values $f(x, y, t_i)$ with $i = 1, ..., m$ and $t_i \in [a, b]$. The implicit assumption will be that $f(x, y, t)$ is continuous on a compact set, which is reasonable in view of the fact that video sequences represent the motion of natural objects in a natural scene (as in video conferencing or surveillance), or natural looking objects in a natural looking scene (as in virtual reality). The construction of the approximating function $g(x, y, t)$ will use function approximation, or learning with connexionist and spiked random neural networks (RNN) [21, 50], using a small number of video frames which act as training samples in supervised learning. It would obviously be impractical to train a different network for each pixel position $(x, y)$. Fortunately, we have observed that it suffices to train a distinct network for pixel blocks which may include 64 or more pixels. the experimental results we present are based on training a distinct network for each block of 64 pixels. Thus only a small number of network weights need to be transmitted from the source to the destination in addition to the frames, to reconstruct consecutive frames from a one out of $S$ received frames. Thus one can achieve siginificantly increased compression ratios of close to $S : 1$, with little additional overhead in the amount of data which is transmitted.

When the objective is to provide good quality end-to-end video transmission through a complex network where frames may be lost or arrive "late", the additional compression ratio achieved by the method expresses the amount of information which may be lost or which may be unavailable to the receiver while the receiver reconstructs a good quality video sequence at the original frame rate. In this case, our method demonstrates that acceptable video sequences can be reconstructde at the receiver when only one out of $S$ frames are in effect received.

[1] Clearly, there are other applications (e.g. a close-up view of a flying aircraft) where such an approach would be much more difficult, unless we could actually model aircraft dynamics in a video scene using the function $f$.

## 1.1 Related Work

Not much work has been done on reconstructing smooth video sequences from a small sample of correctly received video frames. Most of the related work is in image compression research, which generally addresses the 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 [30]. 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 [23] 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 ([35]), the computation overhead still exists. Recently, novel approaches have been introduced based on pyramidal structures [1], wavelet transforms [45], and fractal transforms [25]. These and some other new techniques [29] 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 [36] 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. Neural techniques for image compression have been reported in [6, 9, 11, 12, 24, 31, 27, 33, 32, 34]. Several researchers have used the Learning Vector Quantization (LVQ) network [28] 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 [32]. A Kohonen net method for codebook compression is demonstrated in [34]; 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 [31]. 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 [24], 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 [27], 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 [33], the authors use a backpropagation based nested training algorithm to do compression. Motion detection and prediction are key issues when one deals with moving images. Motion prediction provides for a great deal of the compression in the MPEG standard. MPEG can code the blocks in a frame in terms of motion vectors for the blocks in the past and/or future frames. To perform motion must be estimated using Exhaustive searches which consider all possible motion vectors yield good results. However for the cost of such a search becomes prohibitive and heuristics must be used. This also raises the problem that full motion prediction cannot be performed in real time since it requires the future frame to be known in advance. One should also note that the MPEG standard does not specify the method of motion compensation to be used. 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. 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, for instance in trying to estimate the motion of a human being in which motion vectors differ over subsets of the picture.

## 2 Video Reconstruction with Neural Networks from Temporally Subsampled Frames

In recent work [51] we have demonstrated that it is possible to drop entire frames from a video sequence during compression only to reconstruct them upon decompression. By dropping all but one in S frames of a video sequence, it should be possible to gain a factor of S in the compression ratio. Of course, the challenge is to design a method for reconstructing an accurate rendering of the missing frames from the few frames which are received.

The method for reconstructing frames proposed here uses neural networks to interpolate the missing frames on a pixel by pixel basis. It should be noted that this does not constitute a simple reduction in frame rate as many compression schemes use when/if dropping frames, since in our proposed approach each receiver actually obtains a frame rate which matches the original frame rate of the video sequence. In fact, the receiver frame rate can even be higher than the original frame rate. This concept is outlined in Figure 3 for the case where the subsampling ("downsampling") is voluntary. If the downsampling were accidental, then it would be taking place within ythe transmission medium.

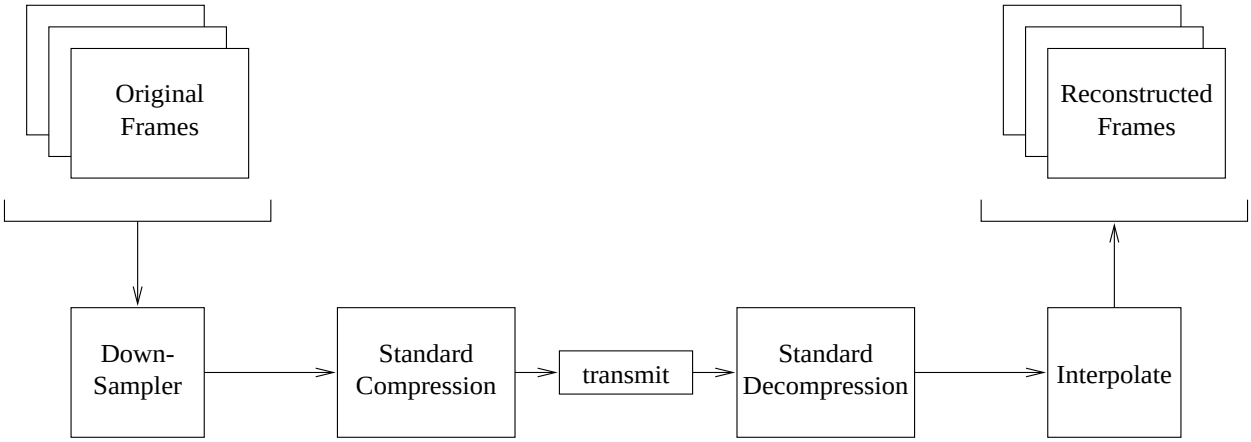



Figure 3: Block diagram for the proposed video reconstruction scheme.

Two neural network based interpolation methods, and cubic spline interpolation, will be used for the interpolation. They will also be compared to a simple linear interpolation which can be viewed as a baseline. The two neural networks considered are a conventional connexionist neural network trained using the backpropagation learning algorithm [36], and the Random Neural Network (RNN) [18, 19, 20] with a feed-forward architecture and its associated learning algorithm [21]. The RNN has previously been used as an optimizing to another networking problem, where the issue was to set up minimal multicast trees [55] for one-to-many or many-to-many communications.

Video can be thought of as a sequence of two dimensional mappings, where each mapping represents a frame and is defined by the function: $h : Z \times Z \rightarrow \{0, 1, ..., 2^k - 1\}$ where $Z$ is the set of whole numbers representing the spatial coordinates of the image, and $k$ is the number of bits used to represent the color and intensity of the pixel, or (in the case of gray-scale images) the intensity of the pixel. This representation underlies many current video compression techniques in which spatial compression is applied to portions of the frame, and the temporal aspect of the sequence is addressed by using motion vectors which are essentially spatial pointers with a temporal reference. Instead of examining a video sequence as a series of 2D spatial mappings, we will instead consider it as a subsample of 3D temporal functions $f(x, y, t)$, representing each pixel $(x, y)$ as it changes through time $t$. If $F$ is the number of frames per second for the video sequence, the $i-th$ frame provides $f(x, y, \frac{i}{F})$ for $(x, y) \in Z$, $i = 1, 2, ....$

The subsampled video sequence only includes one out of every $S$ frames, so that it is composed of only the values $f(x, y, \frac{i}{FS})$, $(x, y) \in Z$, $n = 1, 2, ...$, of the function $f(x, y, t)$ for $t \in [0, \infty]$. To reconstruct the video sequence we construct an approximating function $g(x, y, t)$ which based on the frames points $f(x, y, \frac{i}{FS})$. We then subsample $g(x, y, t)$ at the rate of $F$ frames per second. This yields the discrete function $g(x, y, \frac{i}{F})$ which approximates the original pixel function $f(x, y, \frac{i}{F})$. This operation is performed pixel by pixel; however the approximating function or neural network is constructed on the basis of 64 pixel blocks, rather than distinctly for each individual pixel. The total compression ratio resulting from this scheme is therefore the level of compression which may already be contained in the sequence $f(x, y, \frac{i}{F})$ obtained by some image compression technique, multiplied by the subsampling rate $S$.

## 2.1 Construction of the Approximating Function $g(x, y, t)$

We will consider three distinct approaches to construct the approximating functions $g(x, y, t)$: cubic splines, a conventional connexionist network with backpropagation learning, and the RNN (Random Neural Network) with its learning algorithm. These will be compared with each other and with a simple linear interpolation, with respect to both video quality and compression ratio. Comparisons will also be made with the case when all the video sequences is transmitted and received, so that no interpolation is neded (i.e. the case with $S = 1$). When we compare any video stream for some value of $S$ with the case for $S =$, the video quality expressed as the average Peak-Signal-to-Noise-Ratio over all the frames in the whole video sequence considered, will tell us how much the video sequence has been deteriorated by dropping frames and then reconstituting them by interpolation. The compression ratio will provide us with information about how much data is actually received out of the whole data contained in the original video sequence.

In cubic spline interpolation, the original function is approximated by a cubic spline drawn using the original sequence as control points. Using more control points to interpolate missing frames should capture the pixel dynamics in a video sequence more closely, and result in improved quality in the reconstructed sequence. When we use neural networks to interpolate a sequence, we avoid making assumptions regarding the way in which a pixel changes over time. Instead, one can simply assume that the neural network will "learn" to correctly reconstruct missing pixels based on the information given in the control frames. A neural network should be better able to interpolate missing pixels than the cubic spline technique because the neural network is capable of learning by example and can therefore learn to reproduce non-smooth interpolations, while he cubic spline interpolation is constrained to produce results in which the pixels smoothly change from one frame to the next. However, even if the neural network interpolates only as well as the cubic splines, neural network techniques can be preferred since the resulting a hardware implementation could operate at higher speeds due to the intrinsically parallel nature of neural networks.

### 2.1.1 Neural Network Structure

The connexionist networks we use are three layered. The input layer contains of four neurons, one for each of the four control points in third order interpolation. The number of neurons in the output layer depends upon the amount of subsampling, and is selected to be $3 \times (S - 1)$. The number of hidden neurons used is determined empirically as indicated below.

The Random Neural Network (RNN) is significantly different in their operating principle and learning algorithm (se the Appendix) from their connexionist counterparts. However we have used an identical three layer feed-forward topology in both cases. We provide a mathematical description of the RNN and of its learning algorithm in the Appendix. The RNN we use has an input layer containing four neurons, which correspond to the four control points in third order interpolation. The number of neurons in the output layer depends upon the amount of interpolation that the network will be required to perform so that there are $3 \times (S - 1)$ output neurons. Finally, the number of hidden neurons varies, and is determined as described below.

For both neural network types, the appropriate number of hidden neurons was found by training

multiple networks containing different numbers of hidden neurons. The network that performed best was then chosen. Eleven networks (of each type: connexionist and RNN) were trained for each neural network and for each level of interpolation. Each different network has 5, 7, 9, 11, 13, 15, 17, 19, 21, 23 or 25 neurons in the hidden layer. Each neural network was trained using a subset of frames in the Salesman video sequence. Pixels from each frame that was not dropped were used as input to the neural network. The output of the network was then compared to the corresponding pixels in the dropped frames. The difference between the two was used as the error function in the appropriate learning algorithm (i.e. back-propagation in the case of connexionist networks and the RNN learning algorithm).

Training was performed on the Salesman sequence using the first two sets of frames, so that the neural network was first trained using frames one through $3S$, interpolating using frames 1, $S$, $2S$, $3S$ as control frames, then the sequence was trained again using frames $3S$ through $6S$, where frames $3S$, $4S$, $5S$ $6S$ were used as control points. These two frame sets were used in training using learning rates (i.e. the proportionality constant used to relate the derivative of the error function with respect to the weights, to the change in the value of the weights) of 0.3, 0.1 0.01. The learning rate was initially set to 0.3, and then progressively reduced to 0.1, and then further reduced to 0.01. The learning rate was reduced whenever several epochs of training did not result in a reduction of the error, i.e. in an improvement of the interpolation of test sequence which the network had not "seen" before.

## 2.2 The Degree of Subsampling

For any lossy video compression scheme (i.e. a CODEC), it is possible to generate a performance curve which plots Quality versus Compression Ratio. When one uses subsampling and interpolation on top of the CODEC, one can generate a family of curves, where each curve corresponds to a given level of subsampling. If subsampling and interpolation are a viable approach, then curves for certain values of $S > 1$ should provide greater video quality at a given compression level than original the CODEC with no subsampling ($S = 1$). This would indicate that a better sequence could be generated at a given compression ratio by using subsampling than could be generated at the same compression ratio by the original CODEC.

This empirical observation would also yield a convenient method for determining how much subsampling is "too much". If there is a value $S_{max}$ beyond which the Quality/Compression curve for a larger $S$ is worse than that for $S_{max}$, then $S$ should not be set above $S_{max}$. Our experiments with real sequences indicate that there is significant overlap in the Quality/Compression curves for different values of $S$. Thus for a portion of the compression ratio range, a smaller value of $S$ may be better, while for another range a larger value is to be preferred. Thus subsampling could, in principle, be tuned to the best value of $S$ if one knows what compression ratios one would like to attain.

In all the experiments conducted, we have found that with the exception of the slight performance differences, all third order interpolation methods resemble one another in terms of the shape Quality/Compression curves. Thus, to avoid training and testing a large number of neural networks with a variable number of hidden layers, we have selected the appropriate maximum value of $S$ using cubic spline interpolation.

To determine $S_{max}$, curves were generated using spline interpolation with values of $S$ from 3 to 10. From the Quality/Compression curves we obtained in this it was found that there was little added performance gain to be had by taking an $S$ larger than 6. Furthermore, any gains obtained beyond this value, may be outweighed by the increased buffering (and therefore latency) that would be incurred at the receiver.

# 3 Experimental Evaluation

In this section we evaluate the proposed interpolation schemes from an experimental standpoint. the resulting traffic characteristics, and the video quality.

The traffic characteristics of the video sequences are plotted on the right-hand-side of Figures 4 and 5 for sequences which are initially compressed using JPEG and using our ANVC method [46]. We provide the bit-rate for the video sequences over a five second time period without subsampling (Left) and with subsampling (Right) in both Figures 4 and 5.

The experimental data for the bit rate (in bits per second) for the JPEG compressed Miss America sequence presented in Figure 4 (Left) without subsampling ($S = 1$), shows that the bit-rate is nearly constant over time; the subsampled and sequence (Right) also produces a constant bit-rate and the increased compression ratio (98.1) is nearly identical to 3 times the ratio for the non-subsampled video sequence.

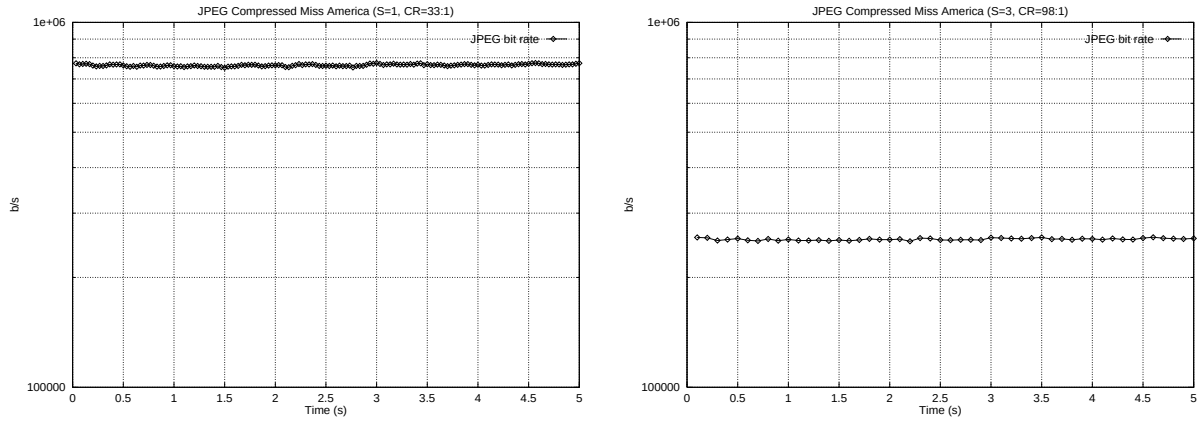



Figure 4: Bit-rates versus time for non-subsampled (Left) and sub-sampled with $S = 3$ (Right) Miss America video sequence, when each frame is initially compressed using JPEG. Notice the constant bit-rates and the quasi $3:1$ compression achieved via subsampling.

The bit rate (in bits per second) for the compressed Miss America sequence, where frames are initially compressed using Adaptive Neural Video Compression (ANVC) [47] is presented in Figure 5 (Left) without subsampling ($S = 1$). The bit-rate for the subsampled sequence (Right) also produces a time-varying bit-rate which closely follows the time variations observed for the case $S = 1$. Despite the value of $S = 3$, the increased average compression ratio (117.1) is only just over 2 times the compression ratio already obtained for in the original (not subsampled) video sequence.

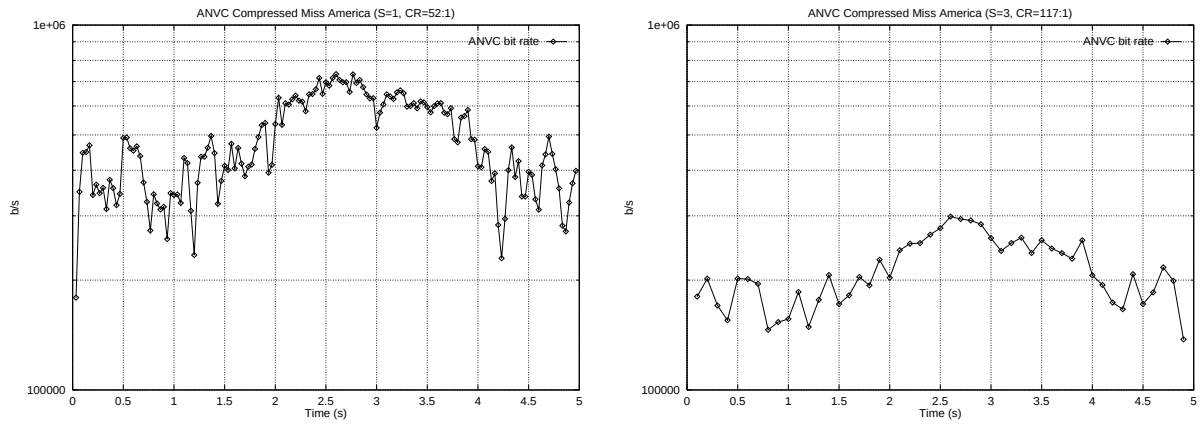



Figure 5: Bit-rates versus time for non-subsampled (Left) and sub-sampled with $S = 3$ (Right) Miss America video sequence, when each frame is initially compressed using Adaptive Neural Video Compression (ANVC) [47]. Notice the time varying bit-rates and the quasi $2:1$ average compression achieved via $S = 3$ subsampling.

These results show that subsampling can provide siginificant reductions in overall traffic, while the impact of this reduction on ATM Quality of Service can be seen in the results shown below.

### 3.1 ATM Cell Loss Rate Reduction

ATM is an important technology for the transmission of multimedia and is used in the transport layer of a wide variety of communication systems. It is therefore a good framework for the evaluation of the impact of the adaptive compression schemes we have suggested on network performance. An important performance or Quality of Service metric in ATM is the cell loss rate. ATM uses switches with finite buffer capacity which can lead to cell loss when traffic rates are high or when traffic is bursty. Therefore in this section we report results which compare the cell loss rate in ATM switches before and after compression using the methods we have presented in this paper.

In order to carry out the evaluation, we have simulated an output buffer queue in an ATM switch. ATM uses 53 octet cells with 5 octet headers. We have varied the output buffer rates (in cells/second) in order to examine the impact of the buffer's output speed on the cell loss rate. Two different but typical values of buffer size (1024 cells and 2048 cells) were chosen. The results are plotted on Figures 6 and 7.

Both for JPEG and ANVC compression, we have considered the case with no subsampling (S=1) and with S=3. The curves on the left-hand-side in both figures refer to S=3, while those on the right-hand-side are for S=1. We see that for the the buffer sizes considered, B has very little impact on the cell loss rates. Similarly, the method used to compress successive frames also has little impact on the resulting cell loss rates. On the other hand, as would be expected, buffer output speed has a significant effect on the loss rate.

We clearly see that the most important effect in reducing cell loss rate is the subsampling. When we go from S=1 to S=3, the cell loss rate obtained for an output speed of 250,000 cells/sec

(roughly 12.5 Mbytes/sec) is approximately the same as the cell loss rate for a speed of 80,000 cells/sec (roughly 4M bytes/sec). Thus subsampling S=3 allows us to obtain an equivalent Quality of Service for much slower buffer service rates (and therefore a much slower network). In the sequel we will see that this reduction in cell loss rates at the same speed (or constant cell loss rates at slower speeds) is not made at the expense of video quality.

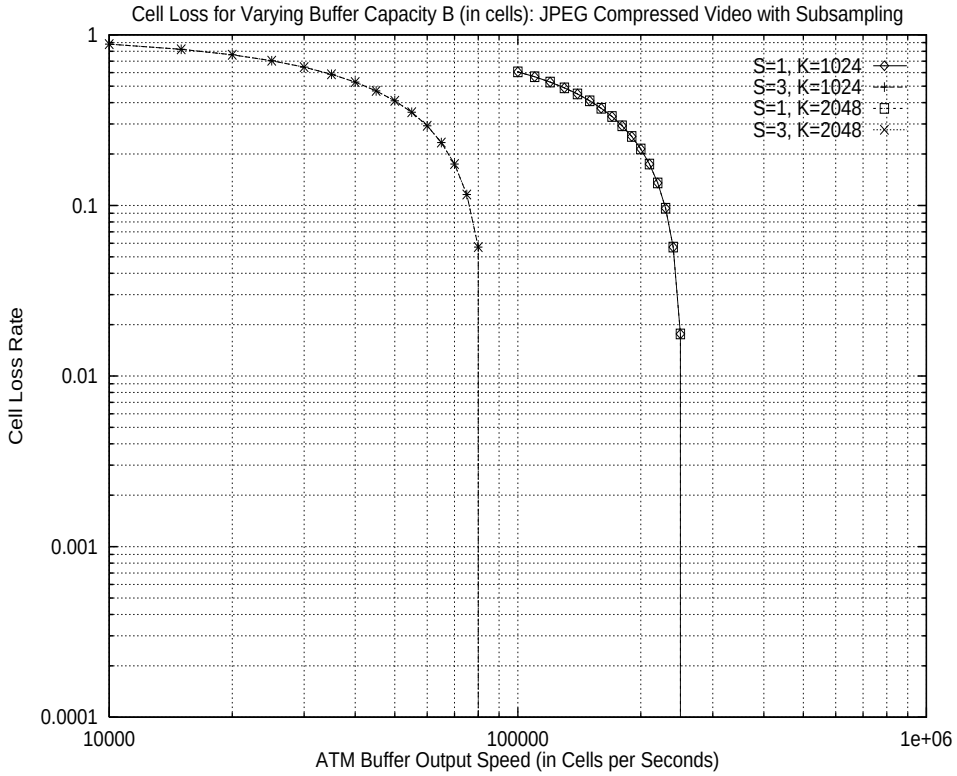



Figure 6:

## 3.2 Video Quality and Compression Ratios for Interpolated Sequences

The video quality, which is determined by the difference between the interpolated video sequence and the original sequence before it is subsampled, cannot be considered in isolation. It must be considered in conjunction with the resulting compression ratio. Therefore the performance measure commonly used is the Quality-versus-Compression-Ratio (QvsCR) characteristic, which plots the observed Peak-Signal-to-Noise-Ratio (PSNR) in decibels (db), averaged frame by frame for all the test sequence, against the measured average compression ratio for the same video sequence.

Typical QvsCR curves are shown in Figures 8 – 10. Our evaluations cover the RNN and the connexionist network based interpolation, as well as the cubic spline and linear interpolation methods. Although we have tested these techniques for arbitrary video sequences obtained with a camera, we will only report results for two well known test sequences: Miss America and Salesman.

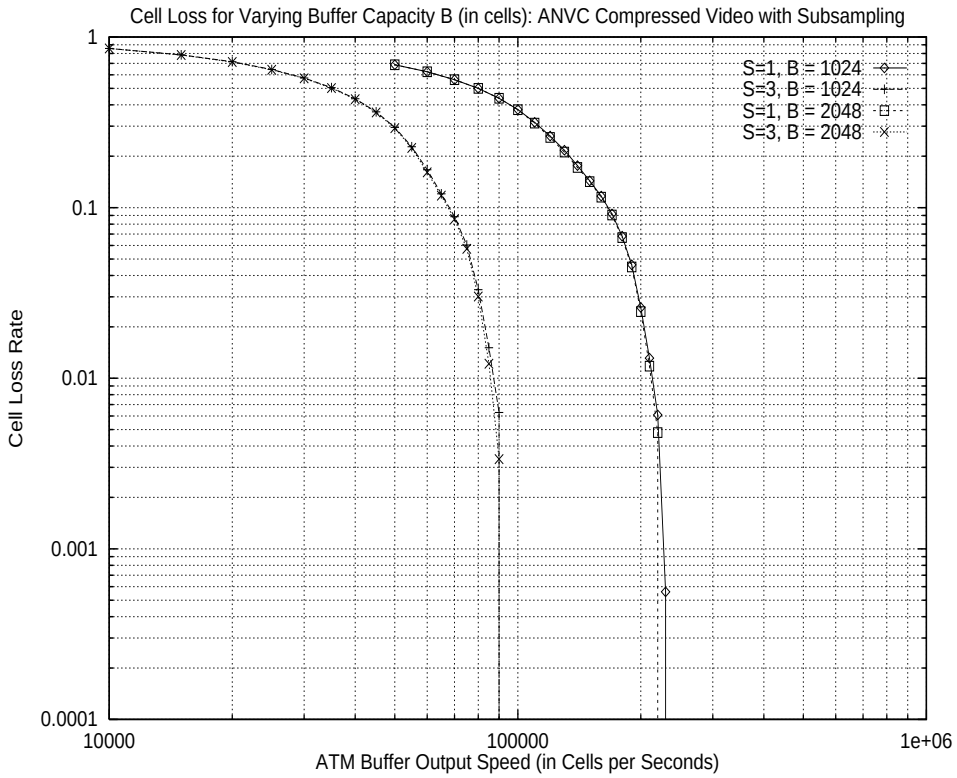



Figure 7:

In order to base our results on viable video compression technqiues, we subsample and then interpolate sequences which have been previously compressed with three different compression algorithms: JPEG [52, 53], H.261 [54], and Adaptive Neural Video Compression [46, 47, 48] which is a Neural Network based video compression method inspired by the work on feature extraction for still image compression by Sonehara *et al* [39]). video sequence and compression technique was tested by varying the degree of subsampling, between $S = 3$ and $S = 6$, yielding 96 different Quality versus Compression Rate (QvCR) curves. Additionally, each set of curves for the four interpolation methods considered was compared to the QvCR curve for $S =$ when no subsampling is carried out.

From Figures 8 – 10, we see that at lower levels of subsampling ($S = 3$) the cubic spline method and the RNN are nearly equivalent. However, for higher levels of subsampling ($S = 6$), the trained RNN outperforms both the cubic spline and the connexionist interpolation methods. In each of these figures four distinct curves are reported: they correspond to the different interpolation methods. Variations in compression and quality levels in each of the curves result from adjusting the basic tradeoffs of the original compression method (i.e. before subsampling and interpolation). For JPEG compression, we fix the compression level and allow the image quality factor (expressed as a target Peak-Signal-to-Noise-Ratio) to be varied, resulting in variations in quality and compression ratios. In the ANVC, we adjust the threshold parameter used to determine motion so as to vary video quality and compression ratio. For H.261, the target compression ratio is varied and as a result the image quality varies as well.

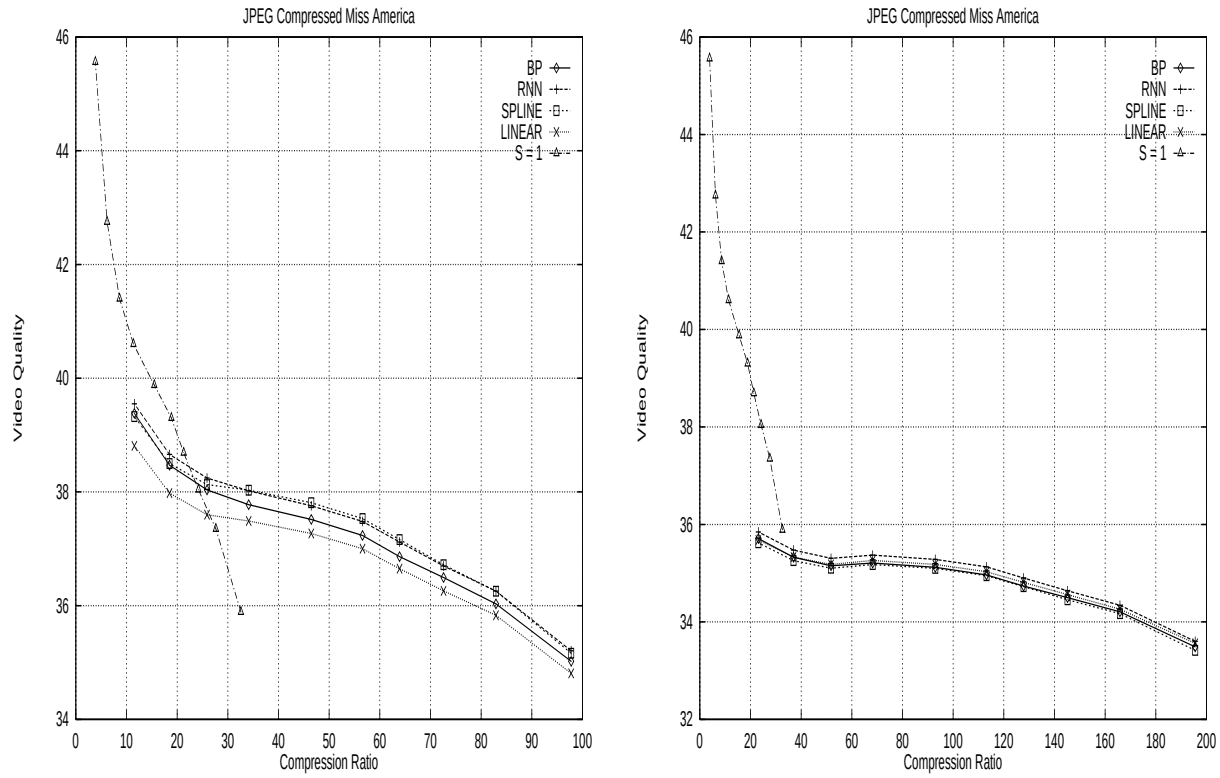



Figure 8: Comparison of interpolation methods for the JPEG compressed Miss America sequence (S=3, Left), (S=6, Right).

### 3.3 Impact of Interpolation on Video Quality

As mentioned earlier, each point on the curves in Figures 8 – 10 is obtained from the average compression value (x-axis), and the average Peak-Signal-to-Noise-Ratio (y-axis), over the whole video sequence being tested. Each of these figures shows the QvCR curves with *no subsampling* for $S = 1$, and with subsampling $S > 1$ and interpolation.

We observe that despite the drop of a significant proportion of frames (2 dropped out of 3 for $S = 3$, and 5 dropped out of 6 for $S = 6$), the video quality as measured by the PSNR drops by at most $10 - db$ with respect to the case for $S = 1$. The largest drop is observed for JPEG compressed video (Figure 8), while in the other cases the drop varies between approximately $4 - db$ to $6 - db$. Note that increases in compression ratio are typically from 5 to 10 times. This small loss in video quality in exchange for massive compression ratio increases is significant. scheme is an excellent approach for achieving good quality video at the receiver, when a significant fraction of frames are dropped or lost. On the other hand, the approach we propose in this paper is also a relatively simple way to obtain increase in the compression ratio, by deliberately dropping frames and then building interpolated frames at the receiver. As such it can be used as a compression method, or as an end-to-end video quality enhancer in the presence of lost or delayed frames, or both.

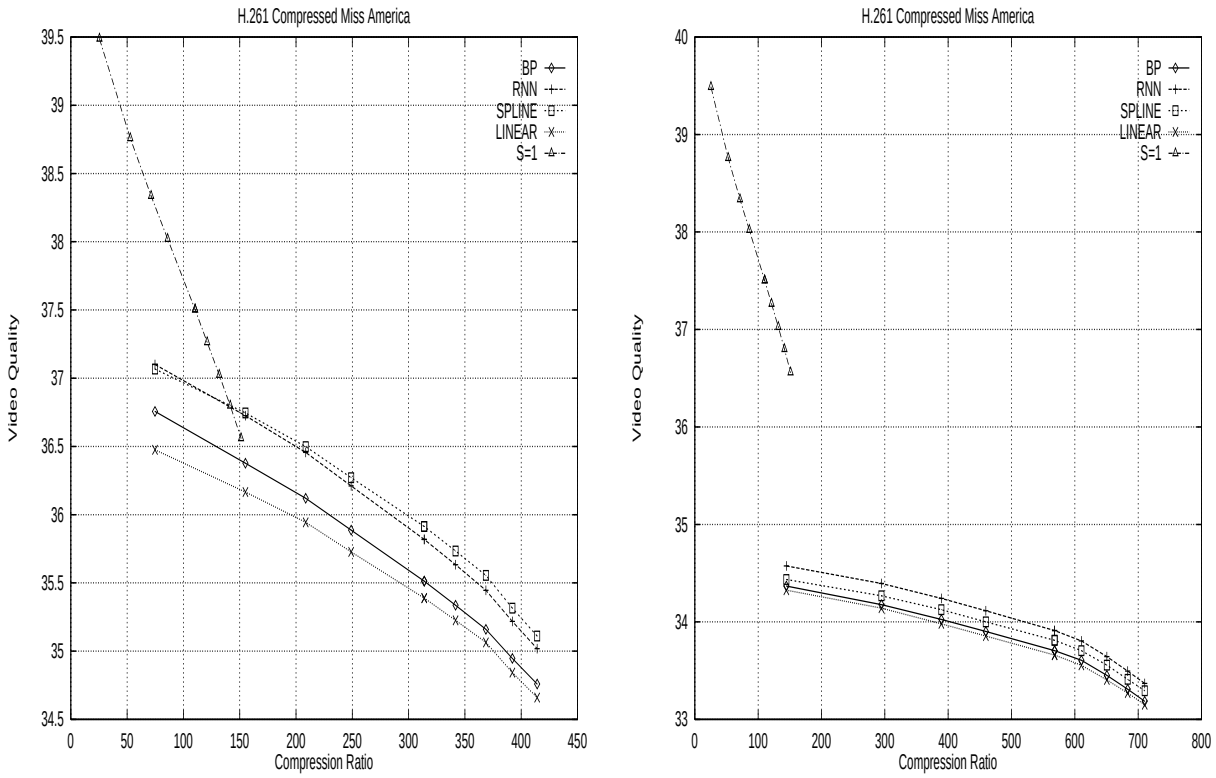



Figure 9: Comparison of interpolation methods for the H.261 compressed Miss America sequence with S=3 (Left) and S=6 (Right)

### 3.4 Comparison of the RNN and the Connexionist Network

In the experimental results, it is seen that the performance of RNN interpolation differs significantly from that of the connexionist network for the MISS AMERICA sequence. This difference is much less noticeable for the SALESMAN sequence. Both types of networks were trained using the first portion of the Salesman sequence and then used on both the entire Salesman sequence and the Miss America sequence.

To test whether the connexionist networks do not generalize as well as the RNN in this application, another set of interpolation networks were trained with the first portion of the Miss America sequence. They were tested on both the entire Miss America sequence and the Salesman sequence. If the RNN outperforms the connexionist network for generalization, one would expect that the RNN and the connexionist networks would perform comparably on the Miss America sequence, while there would be a greater disparity in performance for the Salesman sequence. Figure 11 shows the QvsCR performance for the H.261 compressed MISS AMERICA and SALESMAN sequences (respectively) for $S = 3$; similar results were obtained for other values of $S$ but are not shown here. The figures show the results of interpolating the sequence using the RNN and the connexionist network as trained by both the Salesman and the Miss America sequences. The results for the interpolating the SALESMAN sequence are roughly identical for all four interpolation networks: RNN and connexionist networks with MISS AMERICA and SALESMAN training. In contrast, the results for the interpolation of the MISS AMERICA sequence show that the MISS AMERICA trained neural networks perform *worse* than their SALESMAN trained counterparts, and the connexionist

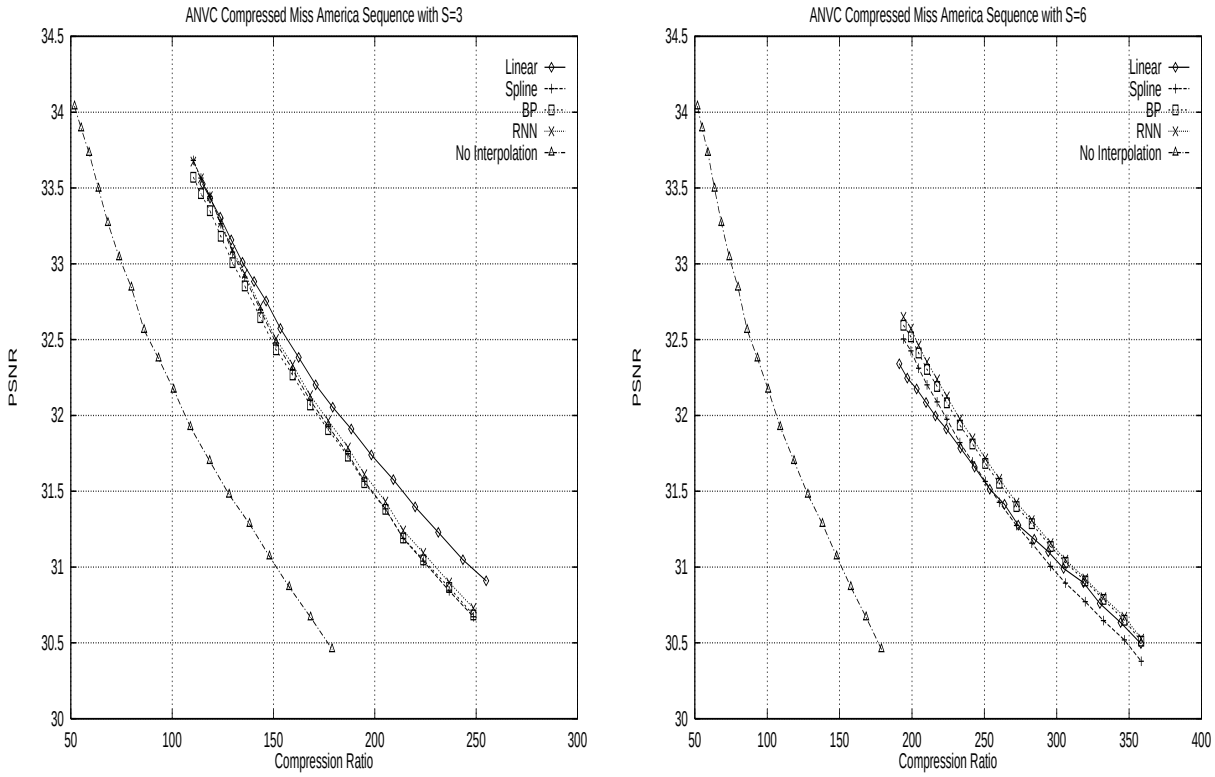



Figure 10: Comparison of interpolation methods for the ANVC compressed Miss America sequence with S=3 (Left), and S=6 (Right).

network is still out-performed by the RNN.

Figure 11 seems to indicate that the problem with the connexionist network is not likely to be related to generalization. Instead, the difficulty may be caused by characteristics of the sequences being used. The characteristics of the sequence may cause it to be both a poor training sequence as well as a difficult sequence to interpolate using a connexionist network. To examine this possibility, histograms of the pixel intensity within each sequence were obtained as shown in Figure 12. We see that in both sequences, the pixel intensities are most frequently low. In the SALESMAN sequence, there is a gradual decrease in frequency at higher intensities, and in the MISS AMERICA sequence the decrease in frequency with intensity is very rapid, with the overwhelming majority of the pixels having an intensity between 25 and 50 (the maximum intensity is 255). It is likely that this reduced range of intensity values causes the MISS AMERICA sequence to be a poor training set for interpolation. The discrepancies in performance between the RNN and the connexionist network indicate that RNN may be more robust to the intensity distributions.

# 4 Conclusions and Further Work

This paper explores the idea of using frame interpolation with a variety of techniques, as a means to build video receivers which reconsitute video sequences having high frame rates with good quality, even when a very significant fraction of frames may have been dropped either at the input of the network, or during transmission. Thus interpolation can be used either as a method for creating high quality video output from unreliable networks, or as compression (subsampling) and decompression (interpolation) methods. We have implemented and tested four interpolation methods to demonstrate these concepts: linear, cubic spline, connexionist neural networks and random neural networks. These interpolation methods were applied to subsampled JPEG, H.261 and ANVC compressed video sequences.

Both traffic characteristics, and Quality versus Compression curves (QvsCR), were studied for two test sequences (SALESMAN and MISS AMERICA), with all four interpolation methods and by varying the degree of subsampling or of "dropping" frames. From these experimental results, several conclusions can be drawn. The first is that the time variations of traffic for subsampled sequences follow closely the similar traffic variations of the non-subsamoled sequences. However, the bit-rates of the subsampled sequnces are – as expected – significantly lower than those for the non-subsamples sequences. However again, the result is not a simple $S:1$ reduction in bit-rates. For JPEG we see that the reductions is just the subsampling factor. However for ANVC [47], which adaptively reacts to motion in the frames, the reduction in bit-rates is apparently less. The reason for this appears to be the greater level of compression which ANVC initially achieves before subsampling.

Experimental results for the QvsCR characteristic depend on the interpolation method used, as well as on the video sequence and its initil compression technique. We see that linear and cubic spline interpolation perform better at lower levels of subsampling, while the neural network techniques perform better at higher levels. This is probably due to the training of the neural networks which gave them some "experience" about the manner in which pixels are likely to change over longer frame sequences. Secondly, it seems that the RNN achieves better performance than a similarly trained connexionist network, and we provided evidence that this performance difference

was not caused by poor generalization by the connexionist network. Rather, our experimental results seem to indicate that the RNN may be more robust to the variations in pixel values of in the sequence equally well on the Salesman sequence. However, further experimental evidence with a variety of sequences will be needed before we can determine why the RNN appears to be a better tool for interpolation. it appears that the performance difference arises due to problems that the connexionist network has with interpolating the Miss America sequence.

Our approach to video reconstruction, which uses pixel-by-pixel interpolation of the dropped frames, can be extended and refined. Future work will focus on enhancing interpolated frames with additional data from the compressed video stream, and one could use intelligent mechanisms that allow the interpolation to focus a greater effort on the most important features within each frame of the video sequence. In recent work [56] we have considered algorithms which allow the dynamic tracking of important regions within video and have applied them to video compression. We could similarly imaging techniques which use different levels of subsampling and interpolation in different dynamically tracked portions of a video sequence.

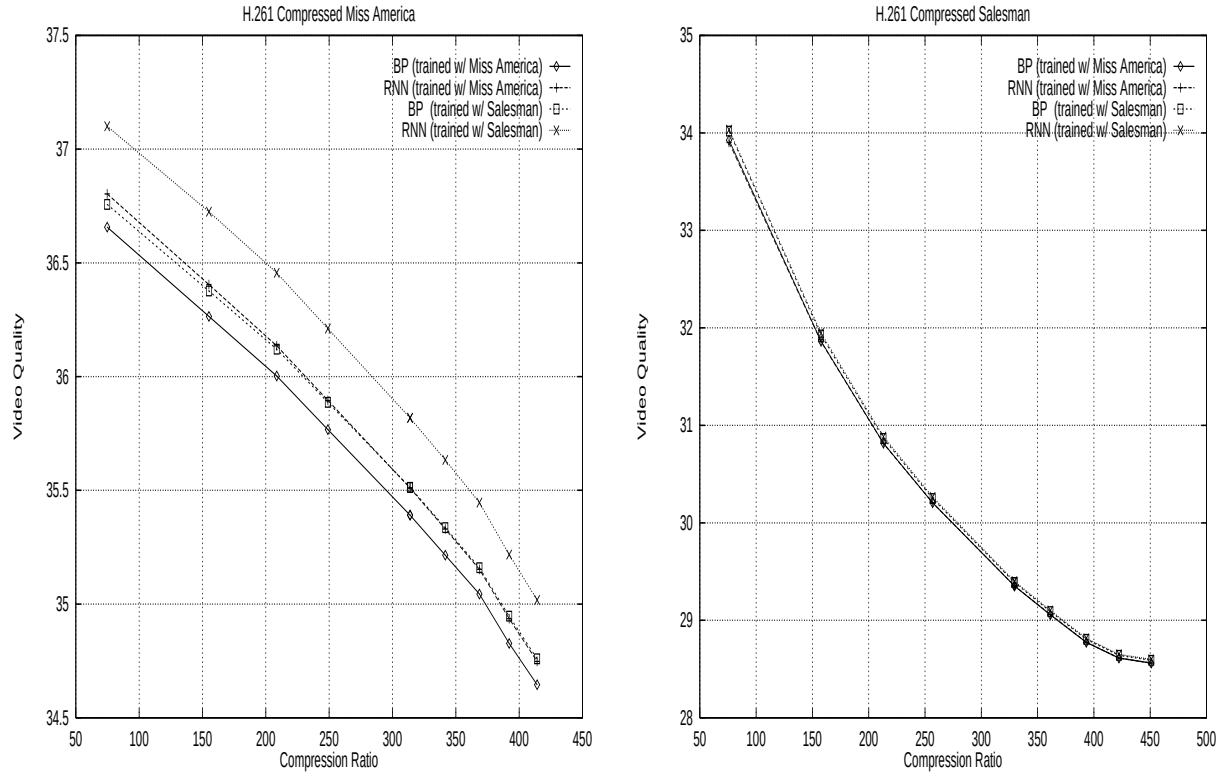



Figure 11: Interpolated (H.261 compressed) Miss America (Left) and Salesman (Right) sequences using various neural compression methods with S=3.

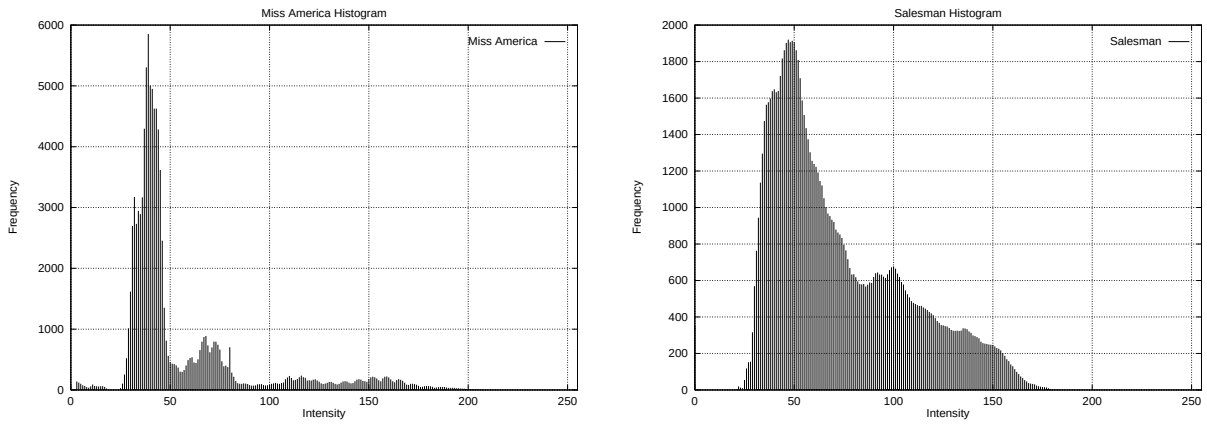



Figure 12: Histogram of pixel intensities in the Miss America (Left) and Salesman sequence (Right).

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'93*, 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 Artificial 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., "Learning in the recurrent random neural network", *Neural Computation*, Vol. 5, No. 1, pp 154-164, 1993.

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

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

[24] 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.

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

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

[27] 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.

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

[29] 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.

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

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

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

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

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

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

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

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

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

[39] Sonehara, N., Kawato, M., Miyake, S., and Nakane, K. "Image data compression using a neural network model," in *Proceedings, International Joint Conference on Neural Networks*, pp. 35–41, IEEE:Piscataway, NJ, (Washington, DC, USA), June 1989.

[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.

[46] Gelenbe, E., Cramer, C.E., Sungur, M., Gelenbe, P. "Traffic and video quality in adaptive neural compression", *Multimedia Systems*, Vol. 4, pp. 357–369, 1996.

[47] Cramer, C. E., Gelenbe, E., Bakircıoglu, H. "Low bit rate video compression with neural networks and temporal subsampling," *Proceedings of the IEEE*, Vol. 84, No. 10, pp. 1529–1543, October 1996.

[48] Cramer, C. E., Gelenbe, E., Gelenbe, P. "Image and video compression", IEEE Potentials, February/March 1998.

[49] Bakircıoglu, H., Gelenbe, E., Koçak, T. "Image processing with the Random Neural Network model", *ELEKTRIK*, Vol. 5 (1), pp. 65-77, 1998.

[50] Gelenbe, E., Mao, Z.H., Li, Y.D. "Function approximation with spiked random networks", *IEEE Trans. on Neural Networks*, Vol. 10, No. 1, pp. 3–9, 1999.

[51] Cramer, C.E. and Gelenbe, E. "Massive neural video compression with temporal subsampling," in *Proceedings of the International Conference on Neural Networks*, pp. 1963–1968, IEEE, IEEE, (Washington D.C., USA), June 1996.

[52] "Digital Compression and Coding of Continuous-Tone Still Images, Part 1, Requirements and Guidelines." ISO/IEC JTC1 Committee Draft 10918-1, February 1991.

[53] "Digital Compression and Coding of Continuous-Tone Still Images, Part 2, Compliance Testing." ISO/IEC JTC1 Committee Draft 10918-2, Summer 1991.

[54] "Video codec for audio visual services at px64 kbits/s." CCITT Recommendation H.261, 1993.

[55] Gelenbe, E., Ghanwani, A., Srinivasan, V. "Improved neural heuristics for multicast routing", *IEEE Journal of Selected Areas of Communications*, Vol. 15, No. 2, pp. 147-155, February 1997.

[56] Feng, G. and Gelenbe, E. "Adaptive object tracking and video compression", *Networking and Information Systems Journal*, Vol. 1, No. 4-5, pp. 371-400, 1998.

## Appendix

## The Random Neural Network

In this appendix we summarize the Random Neural Network Model and of its Learning Algorithm. In the random neural network model [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 in general "recurrent network" model, *i.e.* a network which is allowed to have feedback loops, of arbitrary topology. However we only use it in this work in its feed-forward version.

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 through-

out the computations:

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

It is given by the following result:

**Theorem 1.** *Let $q_i$ denote the quantity*

$$q_i = \lambda^+(i)/[r(i) + \lambda^-(i)] \tag{2}$$

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

$$\lambda^+(i) = \sum_j q_j r(j) p^+(j,i) + \Lambda(i), \qquad \lambda^-(i) = \sum_j q_j r(j) p^-(j,i) + \lambda(i) \tag{3}$$

*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 2 and 3 such that each $q_i < 1$, then*

$$p(k) = \prod_{i=1}^{n} [1 - q_i] q_i^{k_i} \tag{4}$$

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 [21]. It 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:

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

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 $E_k$:

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

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:

$$w_k(u,v) = w_{k-1}(u,v) - \eta \sum_{i=1}^{n} a_i(q_{ik} - y_{ik})[\partial q_i / \partial w(u,v)]_k \tag{5}$$

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

1. $q_{ik}$ is calculated using the input $\iota_k$ 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:

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

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

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

We can now write the vector equations:

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

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

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

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

Notice that

$$\begin{aligned} \partial \mathbf{q} / \partial w^+(u,v) \quad &= \quad \boldsymbol{\gamma}^+(u,v)q_u[\mathbf{I} - \mathbf{W}]^{-1} \\ \partial \mathbf{q} / \partial w^-(u,v) \quad &= \quad \boldsymbol{\gamma}^-(u,v)q_u[\mathbf{I} - \mathbf{W}]^{-1} \end{aligned} \tag{6}$$

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 $\eta$, 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 (6) with the results of (2).
4. Using equation 5 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 $\eta$.

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.