1-20-2006

# Investigation of Different Video Compression Schemes Using Neural Networks

Prem Kovvuri
*University of New Orleans*

# INVESTIGATION OF DIFFERENT VIDEO COMPRESSION SCHEMES USING NEURAL NETWORKS

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
Engineering

by

Prem Kovvuri

B.Eng., Osmania University, Hyderabad, 2001

August 2005

## ACKNOWLEDGMENTS

The completion of the thesis has involved the support of many people. The first and foremost is my thesis advisor Dr. Dimitrios Charalampidis for providing the ideas, suggestions and motivation for my thesis work. He freely bestowed his time, guidance and brilliance beyond his duty. He was a model in dealing the work with great professional responsibility, professionalism and commitment. The amount of knowledge and perception gained from him cannot be quantified. Special thanks to other members of the thesis committee, Dr.Jilkov and Mr. Jovanovich for enriching my skills in academia by sharing with me their intellectual curiosity , professional insight and understanding through their courses.

My special thanks go to Mr.Vijay Kura, a fellow graduate student and a good friend for lending his exquisite programming skills in the beginning. And to my parents, I owe everything; they support me in all that I do to reach new heights. I sincerely dedicate my work to them with all respects.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# ABSTRACT

Image/Video compression has great significance in the communication of motion pictures and still images. The need for compression has resulted in the development of various techniques including transform coding, vector quantization and neural networks. In this thesis neural network based methods are investigated to achieve good compression ratios while maintaining the image quality. Parts of this investigation include motion detection, and weight retraining. An adaptive technique is employed to improve the video frame quality for a given compression ratio by frequently updating the weights obtained from training. More specifically, weight retraining is performed only when the error exceeds a given threshold value. Image quality is measured objectively, using the peak signal-to-noise ratio versus performance measure.

Results show the improved performance of the proposed architecture compared to existing approaches. The proposed method is implemented in MATLAB and the results obtained such as compression ratio versus signal-to-noise ratio are pr

# CHAPTER 1

# INTRODUCTION

Image processing is an important part of modern communications. In general, image processing algorithms require large amounts of memory storage. As a result, the processing time is considerable for processing still images, and even more significant for motion pictures. Thus, the need for image/video compression arises in the modern world of communications in order to get the desired processing times. Various image/video compression techniques have been developed to reduce the amount of data that needs to be processed or transmitted. This results in reduced processing time to achieve the desired targets. There are several challenges faced while developing any image compression technique. Two main challenges include increasing the compression ratio by representing an image with a small number of bits while maintaining an acceptable quality, and increasing the processing speed to meet the real-time application requirements without compromising the image quality. The growing world of communications is continuously increasing the demand for efficient and effective compression schemes[1]-[36]. Thus, the development of image/video compression algorithms is still needed.

Modern digital technology has made it possible to manipulate multi-dimensional signals with systems ranging from simple digital circuits to advanced parallel computers. The manipulation can be divided into three categories namely image processing, image analysis and image understanding. In our case we restrict the focus onto the fundamental concepts of image processing. We further restrict the

study to two-dimensional (2D) image processing as most of the concepts and techniques described can be easily extended to three or more dimensions.

An image defined in the "real world" can be considered as a function of two real variables, say a($x$, $y$) with 'a' being the amplitude (e.g brightness) of the image at the real coordinate position ($x$, $y$) the amplitudes of a given image will almost always be either real numbers or integer numbers. The latter is usually a result of a quantization process that converts a continuous range (say, between 0 and 100%) to a discrete number of levels [34]. In certain image-forming processes, however, the signal may involve photon counting which implies that the amplitude would be inherently quantized. In other image forming procedures, such as magnetic resonance imaging, the direct physical measurement yields a complex number in the form of a real magnitude and a real phase. In this thesis, we will consider amplitudes as reals or integers.

A digital image $a$ [$m$, $n$] described in 2D discrete space is derived from an analog image $a$($x$, $y$) in a 2D continuous space through a *sampling* process that is frequently referred to as digitization. The 2D continuous image $a$($x$,$y$) is divided into *N rows* and *M columns*. The intersection of a row and a column is termed a *pixel*. The value assigned to the integer coordinates [$m$,$n$] with {$m$=0,1,2,...,$M$-1} and {$n$=0,1,2,...,$N$-1} is $a$[$m$,$n$]. In fact, in most cases $a$($x$,$y$)--which we might consider to be the physical signal that impinges on the face of a 2D sensor--is actually a function of many variables including depth ($z$), color ($\lambda$), and time ($t$). In this work, we will consider the case of 2D, monochromatic, static images.

# CHAPTER 2

# IMAGE COMPRESSION AND TECHNIQUES

Image compression attempts to minimize the size, in terms of bytes of a graphics file without degrading the quality of the image to an unacceptable level. The reduction in file size allows more images to be stored in a given amount of disk or memory space. It also reduces the time required for images to be sent over the Internet or downloaded from Web pages [34], [36].

The following example illustrates the requirements for image storage and transmission time. An image of 1024 pixel $\times$ 1024 pixel $\times$ 24 bit without compression would require 3MB of storage and 7 minutes for transmission, utilizing a high speed, 64 Kbit/s, ISDN line. If the image is compressed at a 10:1 compression ratio, the storage requirement is reduced to 300KB and the transmission time drops to under 6 seconds. Seven 1 MB images can be compressed and transferred to a floppy disk in less time than it takes to send one of the original files, uncompressed, over a network. International standards are more portable compared to proprietary high-end solutions. Currently, JPEG is possibly the most popular industry standard technique for the compression of continuous tone images [20].

In this chapter, several compression schemes including lossless and lossy compression methods will be discussed, as a background to the proposed scheme.

## 2 Types of Compression

### 2.1 Lossless Compression

In lossless compression the compression ratio is relatively small since, as the name “lossless” implies, the original data should be reconstructed without any loss. In other words, lossless coding guaranties that the decompressed image is absolutely identical to the image before compression. This is an important requirement for some application domains, e.g. medial imaging, where not only high quality is in demand, but unaltered archiving is a legal requirement. Lossless techniques can also used for the compression of other data types where loss of information is not acceptable, e.g. text documents and program executables [34]-[36].

**Lossless Coding Techniques:**

- Run length encoding.
- Huffman encoding.
- Entropy coding(Lempel/Zev)
- Area coding.

#### 2.1.1 Run length encoding

Run length encoding is a simple method for compression of sequential data. In many data streams, consecutive single tokens are identical. Run length encoding checks the stream for this fact and inserts a special token each time a chain of more than two equal input tokens are found [36]. This special input advises the decoder to insert the particular token *n* times into output stream.

Following is an example of this method:

| Clock | Input | Coder Output | Decoder Output |
|---|---|---|---|
| 1 | A | | |
| 2 | B | A | |
| 3 | C | B | A |
| 4 | C | Ø | B |
| 5 | C | Ø | Ø |
| 6 | C | Ø | Ø |
| 7 | C | Ø | Ø |
| 8 | D | %5C | Ø |
| 9 | E | D | CCCCC |
| 10 | Ø | E | D |
| 11 | Ø | Ø | E |

In the example, there are 9 tokens going into the coder, but just 7 are going out. The effectivity of run length encoding is a function of the number of equal tokens in a row in relation to the total number of input tokens. This relation is very high in two tone images of the type used for facsimile. Effectivity degrades when the input does not contain too many equal tokens. With a rising density of information, the likelihood of two following tokens being the same does sinks significantly, as there is always some noise distortion in the input. Run length coding is easily implemented, either in software or in hardware. It is fast and very well verifiable, but its compression ability is very limited [30]-[36].

### 2.1.2 Huffman coding

This algorithm is based on the fact that in an input stream certain tokens occur more often than others. Based on this knowledge, the algorithm builds up a weighted binary tree according to their rate of occurrence. Each element of this tree is assigned a new code word, whereat the length of the code word is determined by its position in the tree [29]. Therefore, the token which is most frequent and becomes the root of the tree is assigned the shortest code. Each less common element is assigned a longer code word. The least frequent element is assigned a code word which may be twice as long as the input token.

The compression ratio achieved by Huffman encoding uncorrelated data is 1:2. On slightly correlated data, as on images, the compression rate is much higher, the absolute maximum being defined by the size of a single input token and the size of the shortest possible output token (max. compression = token size[bits]/2[bits]). While standard palletized images with a limit of 256 colors may be compressed by 1:4 if they use only one color, more typical images give results in the range of 1:1.2 to 1:2.5.

### 2.1.3 Entropy coding

The implementation of an entropy coder follows with a wide range of modified Lempel/Ziv codings. These algorithms all have a common way of working. The coder and the decoder both build up an equivalent dictionary of metasymbols, each of which represents a whole sequence of input tokens. If a sequence is repeated after a symbol was found for it, then only the symbol becomes part of the coded data

and the sequence of tokens referenced by the symbol becomes part of the decoded data later. As the dictionary is build up based on the data, it is not necessary to put it into the coded data, as it is with the tables in a Huffman coder. This method becomes very efficient on virtually random data. The average compression on text and program data is about 1:2, the ratio on image data comes up to 1:8 on the average GIF image. A high level of input noise degrades the efficiency significantly. Entropy coders are a little tricky to implement, as there are a few tables, all growing while the algorithm runs [28]-[36].

### 2.1.4 Area coding

Area coding is an enhanced form of run length coding, reflecting the two dimensional character of images. This is a significant advance over the other lossless methods. The algorithms for area coding try to find rectangular regions with the same characteristics. These regions are coded in a descriptive form as an Element with two points and a certain structure. The whole input image has to be described in this form to allow lossless decoding.

The possible performance of this coding method is limited mostly by the very high complexity of the task of finding largest areas with the same characteristics. Practical implementations use recursive algorithms for reducing the whole area to equal sized subrectangles until a rectangle fulfills the criteria defined as having the same characteristic for every pixel. This type of coding is highly effective but it bears the problem of a nonlinear method, which cannot be implemented in hardware.

Therefore, the performance in terms of compression time is not competitive, although the compression ratio is.

### 2.2 Lossy Compression

Lossy techniques cause image quality degradation in each compression/ decompression step. Careful consideration of the human visual perception ensures that the degradation is often unrecognizable, though this depends on the selected compression ratio. In general, lossy techniques provide far greater compression ratios than lossless techniques [28]-[36].

In most of the applications we have no need in the exact restoration of stored image. This fact can help to make the storage more effective, and this way we get to lossy compression methods. Lossy image coding techniques normally have three components:

- *Image modelling* which defines the transformation to be applied to the image
- *Parameter quantization* where the data generated by the transformation is quantized to reduce the amount of information.
- *Encoding*, where a code is generated by associating appropriate code words to the raw data produced by the quantizer.

Each of these operations are responsible for the compression. Image modelling is aimed at the exploitation of statistical characteristics of the image (i.e. high correlation, redundancy). Examples are transform coding methods, in which the data is represented in a different domain (for example, frequency in the case of the Fourier

Transform [FT], the Discrete Cosine Transform [DCT], the Kahrunen-Loewe Transform [KLT], and so on), where a reduced number of coefficients contains most of the original information. In many cases this first phase does not result in any loss of information [30]-[33]. The aim of quantization is to reduce the amount of data used to represent the information within the new domain. Quantization is not a reversible operation: therefore, it belongs to the 'lossy' methods. Encoding is usually error free. It optimizes the representation of the information (helping, sometimes, to further reduce the bit rate), and may introduce some error detection codes.

In the following sections, reviews of the most important coding schemes for lossy compression are discussed. Some methods are described in their canonical form (transform coding, region based approximations, fractal coding, wavelets, hybrid methods).

**Lossy Coding Techniques:**

- Transform coding(DCT/Wavelet/Gabor)
- Vector quantization.
- Segmentation and approximation methods.
- Spline approximation methods(Bilinear Interpolation/Regularization)
- Fractal Coding.

### 2.2.1 Transform Coding (DCT/Wavelets/Gabor)

A general transform coding scheme involves subdividing an *N*x*N* image into smaller *n*x*n* blocks and performing a *unitary transform* on each subimage. A unitary transform is a reversible linear transform whose kernel describes a set of complete,

orthonormal discrete basic functions. The goal of the transform is to decorrelate the original signal, and this decorrelation generally results in the signal energy being redistributed among only a small set of transform coefficients. In this way, many coefficients may be discarded after quantization and prior to encoding [35]. Also, visually lossless compression can be achieved by incorporating the HVS contrast sensitivity function in the quantization of the coefficients.

Transform coding can be generalized into four stages:

- Image subdivision
- Image transformation
- Coefficient quantization
- Huffman encoding.

For a transform coding scheme, logical modeling is done in two steps: Segmentation, in which the image is subdivided in bidimensional vectors (possibly of different sizes) and a transformation step, in which the chosen transform (e.g. KLT, DCT, and Hadamard) is applied.

Quantization can be performed in several ways. Most classical approach is to use 'zonal coding', consisting in the scalar quantization of the coefficients belonging to a predefined area (with a fixed bit allocation), and 'threshold coding', consisting in the choice of the coefficients of each block characterized by an absolute value exceeding a predefined threshold [36]. Another way to achieve higher compression factors is to apply a vector quantization scheme to the transformed coefficients. The same type of encoding is used for each coding method. In most cases Huffman coding can be used

successfully. The JPEG and MPEG standards are examples of standards based on transform coding.

### 2.2.2 Vector Quantization

A vector quantizer can be defined as a transform operator $T$ from a $K$-dimensional Euclidean space $R^K$ to a finite subset $X$ in $R^K$ made up of $N$ vectors. This subset $X$ becomes the vector codebook. The choice of the set of vectors is of major importance [11]. The level of distortion due to the transformation $T$ is generally computed as the most significant error (MSE) between the "real" vector $x$ in $R^K$ and the corresponding vector $x' = T(x)$ in $X$. This error should be such as to minimize the Euclidean distance $d$.

An optimum scalar quantiser was proposed by Lloyd and Max. Linde, Buzo and Gray extended it to the case of a vector quantiser. The algorithm they proposed is derived from the KNN cauterization method, and is performed by iterating the following basic operations:

- Subdivide the training set into $N$ groups (called 'partitions' or 'Voronoi regions'), which are associated with the $N$ codebook letters, according to a minimum distance criterion.
- The centroids of the Voronoi regions become the updated codebook vectors.
- Compute the average distortion: if the percent reduction in the distortion (as compared with the previous step) is below a certain threshold, then stop.

Once the codebook has been designed, the coding process simply consists in the application of the *T* operator to the vectors of the original image. In practice, each group of *n* pixels will be coded as an address in the vector codebook, that is, as a number from 1 to *N*.

The LBG algorithm for the design of a vector codebook always reaches a local minimum for the distortion function. A careful analysis of the LBG algorithm's behaviour allows to detect two critical points: the choice of the starting codebook and the uniformity of the Voronoi regions' dimensions [11]. For this reason some algorithms have been designed that give better performances. Initialization of LBG algorithm with random choice of the starting codebook requires a large number of iterations before reaching an acceptable amount of distortion. If the starting point leads to a local minimum solution, the relative stopping criterion prevents further optimisation steps [11].

### 2.2.3 Segmentation and approximation methods

With segmentation and approximation coding methods, the image is modelled as a mosaic of regions, each one characterized by a sufficient degree of uniformity of its pixels with respect to a certain feature (e.g. grey level, texture); each region will have some parameters related to the characterizing feature associated with it. The operations of finding a suitable segmentation and an optimum set of approximating parameters are highly correlated, since the segmentation algorithm must take into account the error produced by the reconstruction region (in order to limit this value within determined bounds). These two operations constitute the logical modelling for

this coding scheme; quantization and encoding are strongly dependent on the statistical characteristics of the parameters of this approximation.

Examples are *polynomial approximation and texture approximation*. For polynomial approximation regions are reconstructed by means of polynomial functions in ($x$,$y$); the task of the encoder is to find the optimum coefficients. In texture approximation, regions are filled by synthesizing a parameterized texture based on some model (e.g. fractals, statistical methods, Markov Random Fields). In polynomial approximations the problem of finding optimum coefficients is quite simple (it is possible to use least squares approximation or similar exact formulations), for texture based techniques this problem is complex [28]-[36].

### 2.2.4 Spline approximation methods (Bilinear Interpolation/Regularisation)

These methodologies fall in the more general category of image reconstruction or sparse data interpolation. The basic concept is to interpolate data from a set of points coming from original pixel data or calculated in order to match some error criteria. The problem of interpolating a set of sparse data is generally ill posed, so some regularization algorithm must be adopted in order to obtain a unique solution. In order to apply this kind of technique to image coding, a good interpolant must be used to match visual criteria. Spline interpolation provides a good visual interpolant, which requires a great computational effort. Bilinear interpolation is easy to implement, while maintaining a good visual quality. Regularization involves the minimisation of an energy function in order to obtain an interpolant which presents some smoothness constraints; it is combined with non-continuities along edges in

order to preserve contour quality during reconstruction. Generally all interpolants computations require the solution of very large linear equation sets, even if related to very sparse matrices. This leads to the use of recursive solution such as relaxation or to the use of gradient descent algorithm.

The use of an interpolation algorithm for image coding techniques such as two source decomposition, where the image is modelled as the sum of two sources; one is the stationary part (it can be considered related to the low frequency content), the second is the residual content coming from non-stationaries such as edges. The first source is coded by means of a prediction scheme that can be one of the previously described interpolants. The second source (the residual) can be coded trough the use of a classical coding method. Two source decomposition is a very effective coding scheme as far as it shows a low tile effect that affects all block coding techniques when compression factors become higher [28]-[36].

### 2.2.5 Fractal coding (texture synthesis, iterated function system [IFS])

Fractal parameters, including fractal dimension, lacunarity, and others have the potential to provide efficient methods of describing imagery in a highly compact fashion for both intra and inter frame applications. Fractal methods have been developed for both noisy and noise free coding methods. Images of natural scenes are used because of the fractal structure of the scene content, but results are reported to be applicable to a variety of binary, monochrome, and colour scenes.

The use of "Iterated Function System" for image compression and synthesis using sets of affine transformations developed for a given image, and a principal

result known as the "collage theorem", intraframe compressions in excess of 10,000:1 and interframe compression in excess of 1,000,000:1 were reported. The collage theorem states that if an image can be covered (approximately) with compressed affine transformations of itself, then the image can be (approximately) reconstructed by computing the attractor of this set of affine transformations.

This convergence was extremely slow, about 100 hours, unless assisted by a person and was presented as an illustration of a scientific possibility, not as a commercial reality. To develop a product that would function in a commercial environment the Iterated Systems had developed the patented technique called the 'Fractal Transform'. The development allowed images to be reduced to a set of fractal equations based on the image being processed, rather than a huge library of pre-calculated, reference, fractal patterns [32]-[34]. Image compression algorithms which are noise free have been reported to be developed from this transform for real time automatic image compression at ratios between 10:1 and 100:1

**2.3 Efficiency and quality of different lossy compression techniques**

The performances of lossy picture coding algorithms are usually evaluated on the basis of two parameters:

- The compression factor (or analogously the bit rate) and
- The distortion produced on the reconstruction.

The first is an objective parameter, while the second strongly depends on the usage of the coded image. A rough evaluation of the performances of a method can be made

by considering an objective measure of the error, like MSE or SNR. For lossy methods described above, average compression ratios and SNR values obtainable are presented in the following table:

| Method | VQ | DCT-SQ | DCT-VQ | AP | SplineTSD | Fractals |
|---|---|---|---|---|---|---|
| **BitRate(bpp)** | 0.8-0.4 | 0.8-0.3 | 0.3-0.08 | 0.3-0.1 | 0.4-0.1 | 0.8-0.0 |
| **SNR(db)** | 36-30 | 36-31 | 30-25 | Image dependent | 36-32 | Image dependent |

**Table 1: Comparison of Compression ratios and SNR values**

### 2.3.1 Comparison of Different Compression Methods

During the last years, some standardisation processes based on transform coding, such as JPEG, have been started. Performances of such a standard are quite good if compression factors are maintained under a given threshold (about 20 times). Over this threshold, artifacts become visible in the reconstruction and tile effect affects seriously the images decoded, due to quantization effects of the DCT coefficients. There are two advantages: first, it is a standard, and second, dedicated hardware implementations exist. For applications which require higher compression factors with some minor loss of accuracy when compared with JPEG, different techniques should be selected such as wavelets coding or spline interpolation, followed by an efficient entropy encoder such as Huffman, arithmetic coding or vector quantization. Some of these coding schemes are suitable for progressive reconstruction .This property can be exploited by applications such as coding of images in a database, for previewing purposes or for transmission on a limited bandwidth channel.

# CHAPTER 3

# IMAGE/VIDEO COMPRESSION USING JPEG/MPEG STANDARD

**Introduction to JPEG**

JPEG stands for “Joint Photographic Experts Group” it is a group of people (experts) working towards establishing the international digital video compression standard for continuous-tone (multi-level) still images which include grayscale and color. JPEG is collaboration between ISO and CCITT committees. For single-frame image compression, the industry standard with the greatest acceptance is JPEG it consists of a minimum implementation (called a baseline system) which all implementations are required to support, and various extensions for specific applications [20]. JPEG compression algorithms in software form a part of a graphics illustration or video editing package. JPEG compression algorithms involves eliminating redundant data, the amount of loss is determined by the compression ratio, typically about 16:1 with no visible degradation. For more compression where noticeable degradation is acceptable compression ratios of upto 100:1 can be employed.

## 3.1 Need for JPEG Compression

For modern applications like the internet, development of video CD’s, video conferencing etc all these applications use graphics and sound intensively and consumes very large amount of physical storage. Example TV-quality full motion video requires 720kb per frame displayed at 30 frames per second to get the motion effect which means one second of motion consumes 22MB of storage, so a standard CD-ROM with 648 MB could only provide 30 seconds of video.

JPEG provides a compression method that is capable of compressing color or gray scale continuous tone images of real world subject such as photograph, still video or any complex graphics that resemble nature subjects. JPEG does not operate on a single algorithm it is built up by various compression techniques which serves as its tools. JPEG allows various configurations of these tools depending on the needs of the user. There are two scheme of compression in JPEG [24]. One is a lossy scheme which means compressed image when decompressed back, isn't the same. The other is a lossless scheme which not loses any of the image data when the compressed image is decompressed back. That is the image looks exactly the same as the original one. But the compression achieved by lossless scheme is not high as lossy, usually about 2:1.

JPEG is developed specifically to discard information that the human eye cannot see. Slight changes in color are not perceived well by the human eye, while slight changes in intensity are. Due to this fact we can see that JPEG does not compress gray scale images as well as colored. usually about 5:1, whereas a colored photographic-quality image maybe compressed from 20:1 to 25:1 without experiencing any noticeable degradation in quality. The exact threshold at which errors become visible also depend on the viewing conditions. The smaller the size of an individual pixel, the harder it is to see an error. So errors are more visible on a monitor 70 or so dots/inch than on a high quality color printout of 300 or more dots/inch.

Thus, most multimedia systems use compression techniques to handle graphics, audio and video data streams and JPEG forms the important compression standard with various compression techniques as building blocks.

### 3.2 JPEG Compression and Decompression flow:

The picture below shows the basic flow diagram of a JPEG algorithm, it tells about the compression and decompression flow in steps [20]-[27].

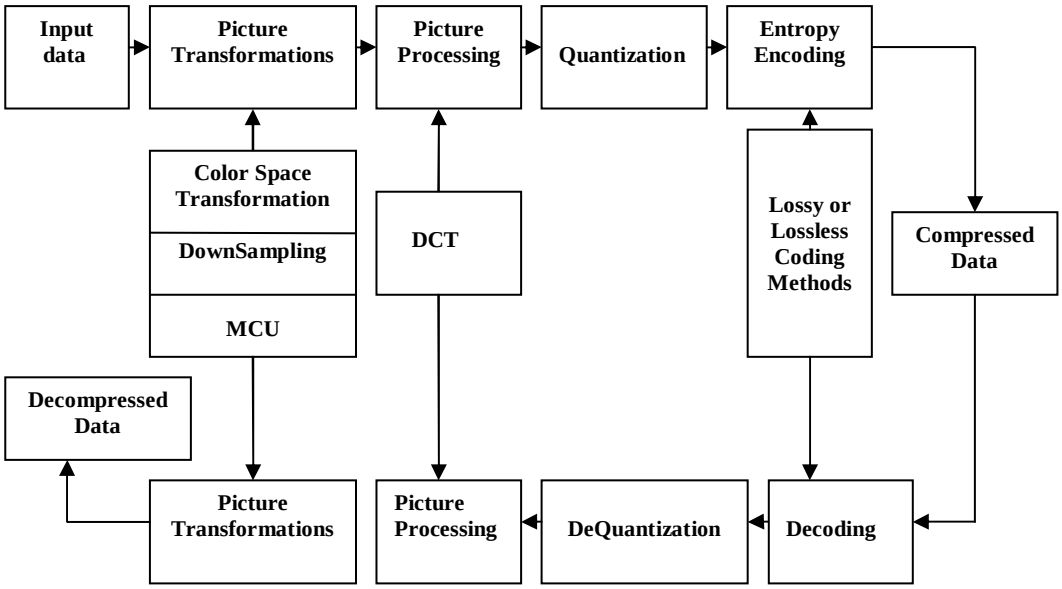



**Figure 1. JPEG Compression and Decompression flow**

**Baseline Lossy JPEG**

Most currently available JPEG hardware and software handles only the Baseline Lossy JPEG (or sequential DCT-based JPEG). The following are the processes discussed in the flow of the algorithm steps:

**Step1: Picture Transformation**

The following activities take place in the picture transformation step:

**ColorSpace Transformation**

This step transforms the image into a suitable colorspace and is not necessary for the proposed scheme because of the gray scale images. For colored images the RGB is transformed into a luminance/chrominance colourspace (YCbCr, YUV etc.). The luminance component is a gray scale while the other two chrominance components are color information, after separating the image into these three components, we will remove more information from the Chrominance (colored) components than the luminance component(optional step). This step increase the compression ratio as it removes unnecessary information in the chrominance components without the human eye detecting the difference.

**Downsample Color Components**

Downsampling reduces the image size by one-half or one-third. It is done by dividing the pixels of each component into groups and for each group we find their average value, and use only one pixel of that average value to represent that whole group. Downsampling is done only to the chrominance components, reducing them by half horizontally and half vertically or no change for the vertical.

**Minimum Coded Unit (MCU)**

An image can be composed of several components, in RGB colorspace we have RED, GREEN and BLUE components and each component is then divided into

data units. In this baseline lossy mode, each data unit is made up of a block of 8*8 pixels. If we processed these data units one component by one component at a time to display the whole image, we call it non-interleaved mode. Frame buffer is required in non-interleaved mode to store all the pixel's values in every component except for the very last one. Together with the values stored in the frame buffer and the pixel's values of the last component, we will be able to determine the actual value of a specific pixel.

Interleaving eliminates the use of frame buffer. To display an image, using interleaved mode, we take a few blocks of data units from each component and display them immediately. We don't wait for the whole picture to be formed in the frame buffer. The picture is slowly built up as the blocks are processed. Interleaved data units of different components are combined into MCU, if all components have the same resolution, an MCU consists of exactly one data unit for each component. The decoder displays the image MCU by MCU.For a set of color components with different resolutions, the MCU is defined interms of frequency of the blocks. According to the JPEG standard, up to four components can be coded using interleaved mode. Each MCU consists of at most ten data units. Within the image, some components can be encoded in the interleaved mode and others in the non-interleaved mode.

**Step 2: Picture Processing**

**Discrete Cosine Transformation (DCT)**

In this stage the uncompressed image samples are grouped into data units of 8*8 pixels and passed to the encoder according to the order defined by the MCU. Then each of the 8*8 pixels' values go through a transformation performed by DCT, using an explicit formula written in terms of the pixel values $f(x,y)$ and the frequency domain transform coefficients, $F(x,y)$.

$$F(u,v)=\frac{1}{4}C_u C_v {}^{7}{}_{x=0}{}^{7}{}_{y=0}(y,x)\cos\left[\frac{(2x+1)u}{16}\right]\cos\left[\frac{(2y+1)v}{16}\right] \qquad (3.1)$$

Where $C_{u,}C_{v}=\frac{1}{2}^{\frac{1}{2}}$ for *u*, v =0*;* otherwise $C_u, C_v = 1$

The output of the transformation will result in the mean value, the DC coefficient is located on the top left corner of the data unit and higher frequency coefficients will be further away from this DC coefficient. Higher vertical frequencies will be represented by higher row numbers where higher horizontal frequencies will be represented by higher column numbers [25].

For reconstruction of the image, the inverse DCT formula is used:

$$(y,x)=\frac{1}{4}\ {}^{7}{}_{x=0}\ {}^{7}{}_{y=0}\ C_u C_v F(u,v)\cos\left[\frac{(2x+1)u}{16}\right]\cos\left[\frac{(2y+1)v}{16}\right] \qquad (3.2)$$

Where $C_{u,}C_{v}=\frac{1}{2}^{\frac{1}{2}}$ for *u*, *v* =0*;* otherwise $C_u, C_v = 1$

when forward DCT is being applied for an image we can see a great reduction on the size of the data. The transformation will result in many zero coefficients and greater

concentration of non-zero values on the upper left corner of the data units. When an inverse DCT is applied to the frequency domain we will get back the initial picture but not a perfect exact reconstruction, as precision will be lost during the rounding off of DCT coefficients from real to integer values (the same thing happens when inverse DCT is applied).

Therefore if Forward Discrete Cosine Transformation (FDCT), as well as the Inverse Discrete Cosine Transformation (IDCT), could be calculated without loss in precision then we will be able to reproduce exactly the same data unit that we started with. This is why DCT is considered a lossy process.

**Step 3: Quantization**

Quantization is used to further reduce the values of DCT coefficients in order to produce more zero coefficients. In Baseline Lossy JPEG the stepsize is varied according to the coefficient location and which color component is encoded [26].

The equation for quantization is:

$$C(v,u) = \frac{[F(v,u)(Q(v,u)/2)]}{Q(u,v)} \tag{3.3}$$

Where $C(v,u)$, is actually the quantized coefficient, $F(v,u)$ is the DCT frequency coefficient, and $Q(v,u)$ is the quantizer stepsize for the pixel $(v,u)$ in the block. The sign indicates a plus for a positive DCT coefficeint, $F(v,u)$, and a minus for a negative DCT coefficient, $F(v,u)$.

The inverse quantizer equation is given as:

$$F(u,v) = C(u,v) * Q(u,v) \tag{3.4}$$

Quantization is also a lossy process. In quantizing an image, the quality factor set, will have direct effect on the amount of Quantization performed. If too much quantization is done to the image, it will cause the final quantized image to look "blocky". Similarly, if too little quantization is performed, it will result in coding useless data (or noise) of the image.

**Step 4: Entropy Encoding**

**Coding Model**

Before actual entropy is performed to the quantized DCT coefficients, the coefficients are rearranged into a one dimensional array using a zig-zag pattern by the code model, with the lowest frequency first and highest frequency last. The zig-zag pattern is used to increase the consecutive runs of zeros for RLE. During this stage the quantized DC coefficient is treated separately from the AC coefficient

**Differential Pulse Code Modulation (DPCM)**

The DC coefficient determines the basic color of a data unit and this value varies slightly between successive blocks. The coding of the DC coefficient is done by Differential Pulse Code Modulation (DPCM), which codes the differential between the quantized DC coefficient of the current block and the quantized DC coefficient of the previous block. The formula for the DPCM code:

$$DPCMcode = C(0,0)_j - C(0,0)_{j-1} \tag{3.5}$$

Where j represents the number of the quantized block being processed. The inverse DPCM returns the current DC coefficient value of the quantized block being

processed by summing the current DPCM code with the previous DC coefficient value of the previous quantized block.

$$C(0,0)_j = DPCMcode_j + C(0,0)_{j-1} \tag{3.6}$$

The DPCM code is represented by the size of the DPCM code followed by the significant value of the DPCM code [20]-[27].

**RLE**

The quantized AC coefficients usually contain a number of consecutive runs of zeros. Therefore RLE is used to encode these zero values.

**Huffman \ Arithmetic Encoding**

Huffman or Arithmetic encoding is used to transform the non-zero AC-coefficients and the DC coefficients into a spectral representation to compress the data even more, the number of bits required depends on the coefficient's value. A non-zero AC-coefficient will be represented between 1 to 10 bits. For the representation of DC-coefficients, a higher resolution of 1 bit to a maximum of 11 bits is used.

### 3.3JPEG Applications

**Baseline Lossy JPEG**

- More for use of storing photograph-like images and naturalistic artworks.
- Due to its great compression efficiency, and permit the ease of exchanging images with widely varying display hardware, it is widely used in the Usenet and World Wide Web.

**Progressive JPEG**

- The advantage of Progressive JPEG is that it allows viewer to see a rough idea of what the actual image looks like and gradually improves the quality. Progressive JPEG is slowly gaining popularity in the World Wide Web because of its advantage, and more and more software are starting to support it including some WWW browser and other programs.

**Motion JPEG (MPEG)**

- Usually used in professional video application areas such as Non Linear Editing Systems (NLE), Digital Disk Recorder (DDR) and Media Servers. Here video compression is used to reduce implementation cost.
- Lossless Motion JPEG is used in areas where video quality is of primary importance such as Digital video compositing, 3D animation and Medical video and photography.

### 3.4 Introduction to MPEG

MPEG stands for “Moving Pictures Exerts Group”, it is a group of people getting together under ISO (International Standard Organization) to generate standards for digital video (sequence of images in time) and audio compression [13]. The compression algorithms developed depends on the individual manufacturers.

MPEG defines a bit stream for compressed video and audio optimized to fit a band width of 1.5Mbps necessary for audio CD’s and DAT’s. The standard is divided into three parts video, audio and systems. The systems part is used to integrate the audio and video streams with proper time stamping to allow the synchronization of the two. MPEG involves in encoding only key frames through the JPEG algorithm (described above) and estimates the motion changes between these key frames. Since minimal information is sent between every four or five frames, a significant reduction in bits required to describe the image results. Consequently, compression ratios above 100:1 are common. The MPEG encoder is very complex and places a very heavy computational load for motion estimation. Decoding is much simpler and can be done by desktop CPUs or with low cost decoder chips. The MPEG encoder makes a prediction about an image and transforms and encodes the difference between the prediction and the image. The prediction accounts for movement within an image by using motion estimation [13], [14]. A given image's prediction may be based on future images as well as past ones, the encoder must reorder images to put reference images before the predicted ones. The decoder puts the images back into display sequence. It takes in the order of 1.1-1.5 billion operations per second for real-time MPEG encoding.

## 3.5 MPEG Compression Standards

There are five MPEG standards that are currently being used and also under further development. Each compression standard is designed based on a specific application and bit rate [13]-[19].

**MPEG-1(Designed for upto 1.5 Mbps):** This standard is based on CD-ROM applications and is popular for video on internet transmitted as .mpg files, level 3 of MPEG-1 is a popular standard for digital compression of audio known as MP3, it is also the standard of compression for video CD.

**MPEG-2 (Designed between 1.5 and 15 Mbps):** this standard is set for digital television set top boxes and DVD compression. It is based on MPEG-1, but designed for the compression and transmission of digital broadcast television. The most significant enhancement from MPEG-1 is its ability to efficiently compress interlaced video. MPEG-2 scales well to HDTV resolution and bit rates, obviating the need for an MPEG-3.

**MPEG-4**: this standard is set for multimedia and Web compression. MPEG-4 is based on object-based compression, similar in nature to the Virtual Reality Modeling Language. Individual objects within a scene are tracked separately and compressed together to create an MPEG4 file. This results in very efficient compression and is very scalable; from low bit rates to very high. It allows developers to control objects independently in a scene, and therefore introduces interactivity.

**MPEG-7:** this standard is currently under development, it is called as the Multimedia Content Description Interface. The objective is to provide a framework for multimedia content that will include information on content manipulation, filtering and personalization, as well as the integrity and security of the content. Contrary to the previous MPEG standards, which described actual content, MPEG-7 will represent information about the content.

**MPEG-21:** this standard is for Multimedia Framework which is under development. MPEG-21 will attempt to describe the elements needed to build an infrastructure for the delivery and consumption of multimedia content, and how they will relate to each other.

### 3.6 MPEG Comparision

All MPEG standards are back compatible meaning MPEG-1 video sequence can be packetized as MPEG-2 or MPEG-4 video. Similarly, MPEG-2 can be paketized as MPEG-4 video sequence. The difference between a true MPEG-4 video and an MPEG-4 paketized MPEG-1 video sequence is that the lower standard does not make use of the enhanced or new features of the higher standard. Both MPEG-2 and MPEG-4 covers a wide range of picture size and picture rates and bandwidth usage, so MPEG-2 introduced a concept called as *Profile@ Level* to communicate compatibilities among applications, example studio profile of MPEG -4 is not suitable for PDA and vice-versa[13]-[19].

The comparison of MPEG's is given in the following table with limitations to MPEG-1 on Constrained Parameters Bitstream (CPB), MPEG-2 on Main Profile at mainlevel (MP@ML) and MPEG-4 on Main Profile at Level 3.

| **MPEG** | **1** | **2** | **4** |
|---|---|---|---|
| Max Bit Rate (Mbps) | 1,86 | 15 | 15 |
| Picture width(pixels) | 352 | 720 | 720 |
| Picture height(pixels) | 288 | 576 | 576 |
| Picture rate (fps) | 30 | 30 | 30 |

**Table 2: Comparison of MPEG**

### 3.7 Work Procedure of an MPEG

An MPEG starts with a relatively low resolution video sequence (possibly decimated from the original) of about 352 by 240 frames by 30 frames/s but with original high (CD) quality audio. The color images are converted to YUV space, and the two chrominance channels (U and V) are decimated further to 176 by 120 pixels.

The basic MPEG scheme is to predict motion from frame to frame in the temporal direction, and then use DCT's (discrete cosine transforms) to organize the redundancy in the spatial directions. The DCT's are done on 8×8 blocks, and the motion prediction is done in the luminance (Y) channel on 16×16 blocks.Given,the 16×16 block in the current frame of coding, we look for a close match to that block in a previous or future frame (there are backward prediction modes where later frames are

sent first to allow interpolation between frames) [15].The DCT coefficients (of either the actual data, or the difference between this block and the close match) are "quantized", which means we divide them by some value to drop bits off the bottom end, many of the coefficients will then end up being zero. The quantization can change for every "macro block" (a macro block is 16×16 of Y and the corresponding 8×8's in both U and V). The results of all of this, which include the DCT coefficients, the motion vectors, and the quantization parameters is Huffman coded using fixed tables. The DCT coefficients have a special Huffman table that is "two-dimensional" in that one code specifies a run-length of zeros and the non-zero value that ends the run. Also, the motion vectors and the DC DCT components are DPCM (subtracted from the last one) coded.

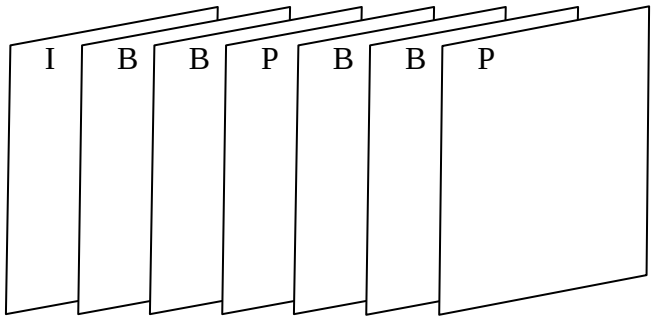
There are three types of coded frames. They are I, P and B. the "I" frames are called as intra-frames, these frames are coded as a still image, not using any past history. The "P" frames are called as predicted frames which are predicted from the most recently reconstructed I or P frame [16], [17]. Each macro block in a P frame can come with a vector and difference DCT coefficients for a close match in the last I or P frames, or it can just be "intra" coded (like in the I frames) if there is no good match. Lastly, the "B" frames which are called as the bidirectional frames, they are predicted from the closest two I or P frames, one in the past and one in the future. We search for matching blocks in those frames, and see which works best. The sequence of decoded frames usually goes like:

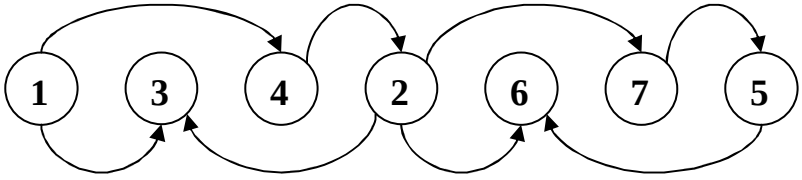
IBBPBBPBBPBBIBBPBBPB...

Where there are 12 frames from I to I this is based on a random access requirement we need a starting point at least once every 0.4 seconds or so. The ratio of P's to B's is based on experience. For the decoder to work, we send the first P before the first two B's, so the compressed data stream ends up looking like:

0xx312645...

where numbers are frame numbers and xx might be nothing (if above is the true starting point), or it might be the B's of frames -2 and -1 if we are in the middle of the stream. We have to decode the I, then decode the P, keep both of those in memory, and then decode the two B's. We display the I while we are decoding the P, and display the B's as we are decoding them, and then display the P as we are decoding the next P, and so on.




**Coding Order**

**Figure 2. Flow of an MPEG**

# CHAPTER 4

# NEURAL NETWORKS

**Introduction to neural networks**

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the biological nervous systems, such as the brain. The key element of this paradigm is the structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurones) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurones. This is true of ANNs as well [2]-[12].

## 4.1 Use of neural networks

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyse. This expert can then be used to provide projections given new situations of interest.

Advantages:

- Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.

- Self-Organisation: An ANN can create its own organisation or representation of the information it receives during learning time.
- Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
- Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

## 4.2 Human and Artificial Neurons

### 4.2.1 How the Human Brain Learns?

In the human brain, a typical neuron collects signals from others through a host of fine structures called *dendrites*. The neuron sends out spikes of electrical activity through a long, thin stand known as an *axon*, which splits into thousands of branches [6]. At the end of each branch, a structure called a *synapse* converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neurons. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes.

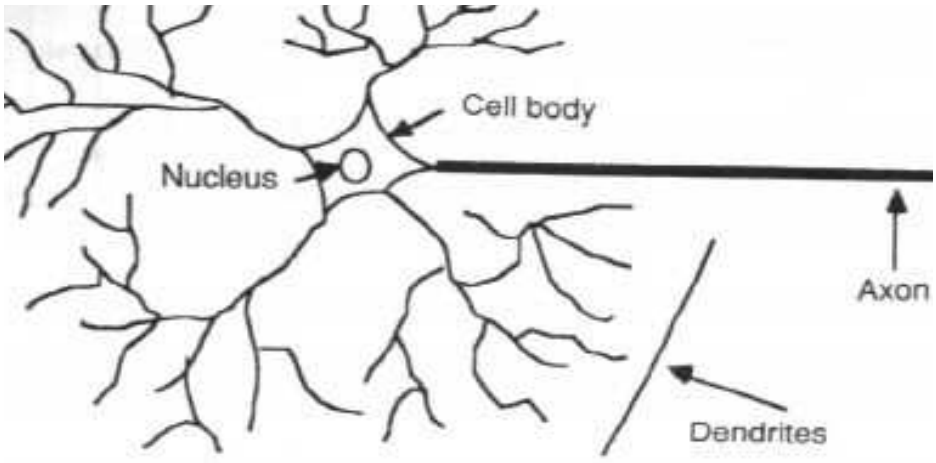



**Figure 3. Components of a neuron**

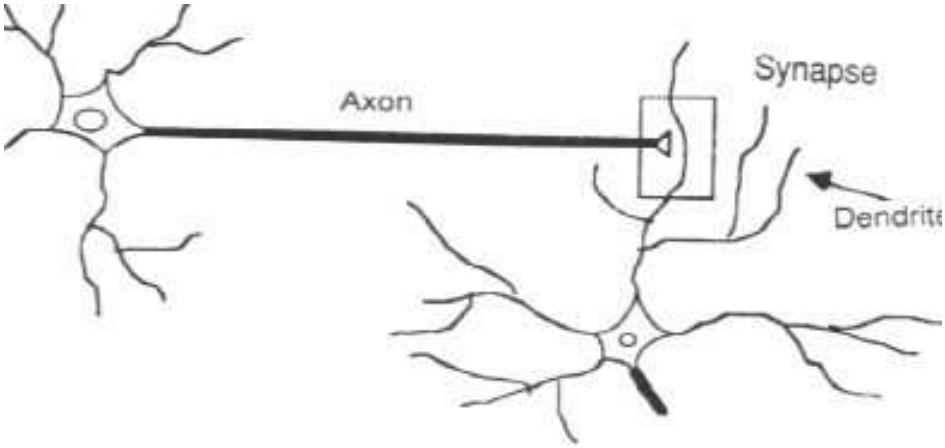



**Figure 4. Synapse**

### 4.2.2 From Human Neurons to Artificial Neurons

By deducing the essential features of neurons and their interconnections. We program a computer to simulate these features [9]. However because our knowledge of neurons is incomplete and our computing power is limited, our models are necessarily gross idealizations of real networks of neurons.

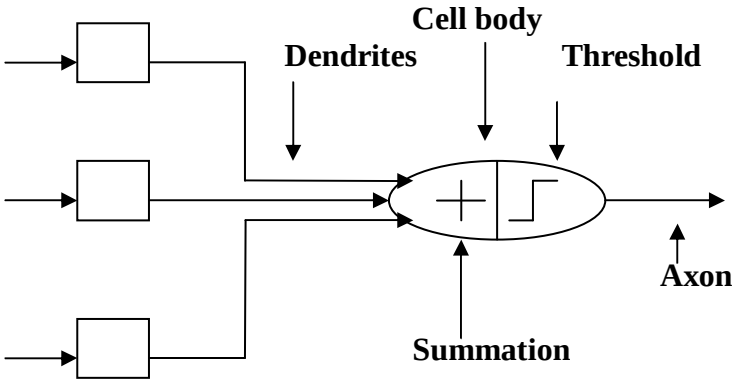



**Figure 5. The neuron model**

### 4.2.3 A simple neuron

An artificial neuron is a device with many inputs and one output. The neuron has two modes of operation; the training mode and the using mode. In the training mode, the neuron can be trained to fire (or not), for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output [10]. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not.

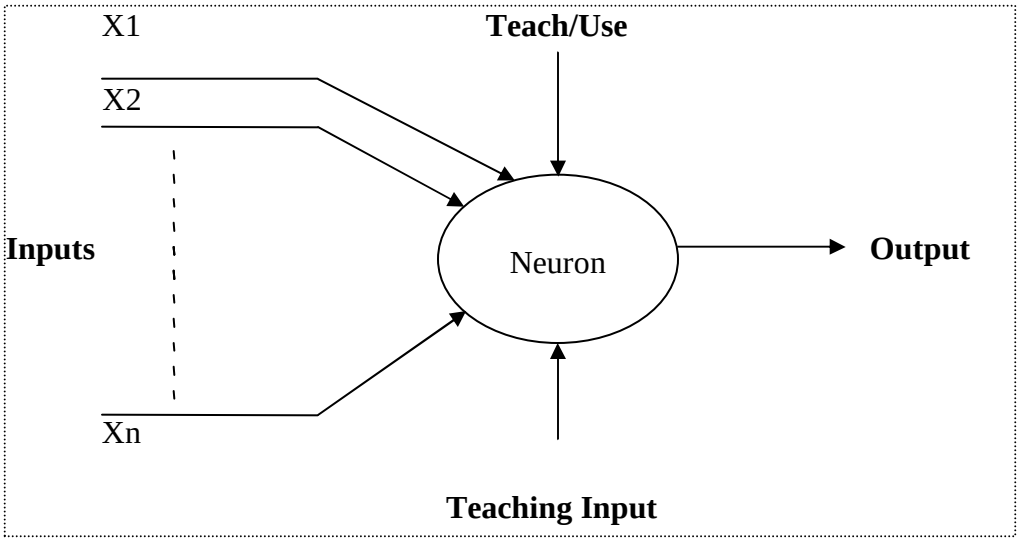



**Figure 6. A simple neuron**

### 4.2.4 A more complicated neuron

A more sophisticated neuron is the McCulloch and Pitts model (MCP). The difference from the previous model is that the inputs are 'weighted', each inputs decision making is dependent on the weight of the particular input. The weight of an input is a number which when multiplied with the input gives the weighted input. These weighted inputs are then added together and if they exceed a pre-set threshold value, the neuron fires. In any other case the neuron does not fire [11].

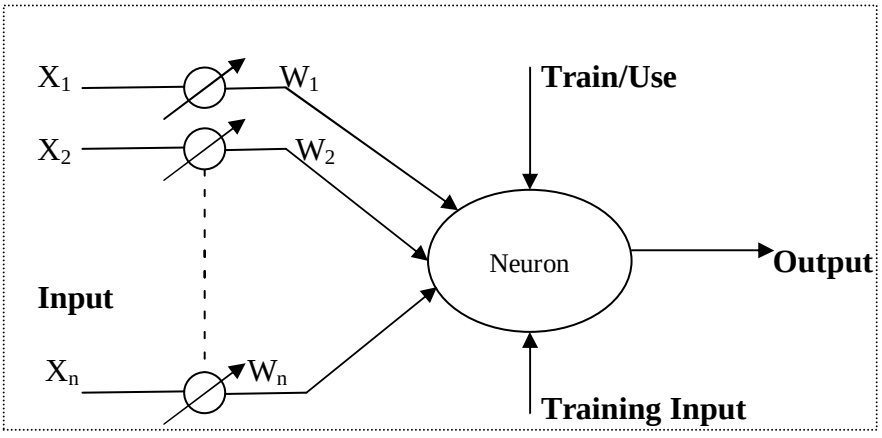



**Figure 7. An MCP neuron**

In mathematical terms, the neuron fires if and only if;

$$X_1W_1 + X_2W_2 + X_3W_3 + ... > T \quad (4.1)$$

The addition of input weights and of the threshold makes this neuron a very flexible and powerful one. The MCP neuron has the ability to adapt to a particular situation by changing its weights and/or threshold. Various algorithms exist that cause the neuron to 'adapt'; the most used ones are the Delta rule and the back error propagation. The former is used in feed-forward networks and the latter in feedback networks.

### 4.3 Architecture of neural networks

#### 4.3.1 Feed-forward networks

Feed-forward ANNs allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs [2]-[12]. They are extensively used in pattern recognition. This type of organisation is also referred to as bottom-up or top-down.

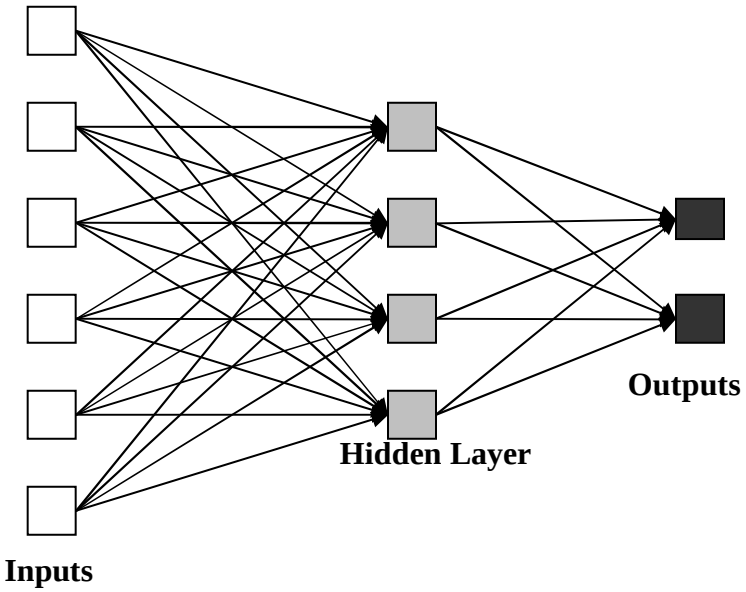



**Figure 8. An example of a feedforward network**

### 4.3.2 Feedback networks

Feedback networks can have signals travelling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, the latter term is used to denote feedback connections in single-layer organisations.

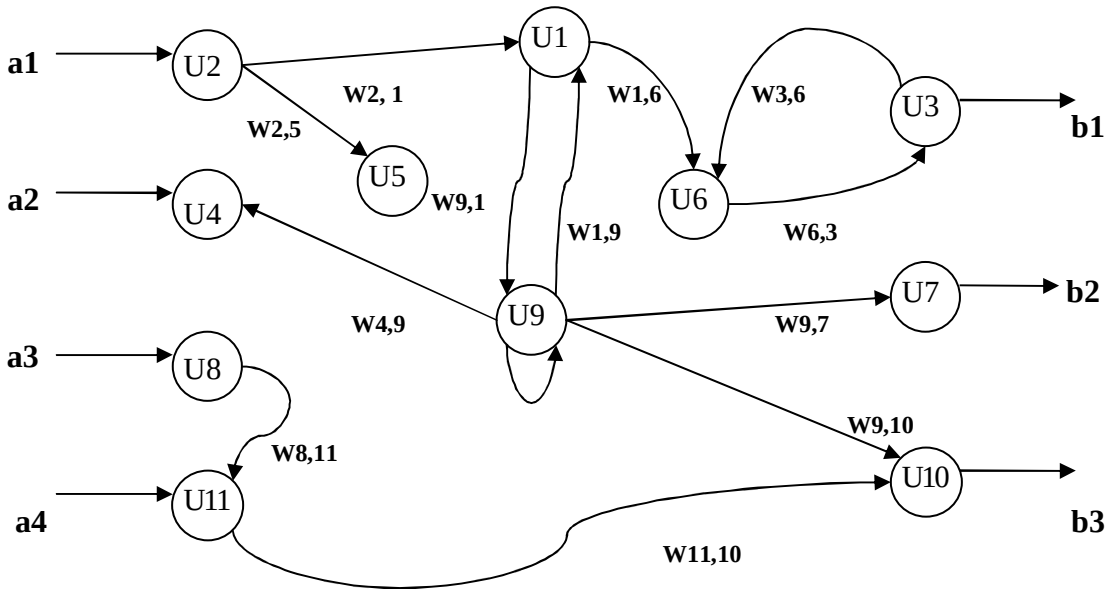



**Figure 9**. **An example of a complicated network**

### 4.3.3 Network layers

The common artificial neural network consists of three groups, or layers, of units: a layer of "input" units connected to a layer of "hidden" units, which is connected to a layer of "output" units.

- The activity of the input units represents the raw information that is fed into the network.

- The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.
- The behaviour of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

The hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents.We also distinguish single-layer and multi-layer architectures. The single-layer organization, in which all units are connected to one another, constitutes the most general case and is of more potential computational power than hierarchically structured multi-layer organizations[2]-[9]. In multi-layer networks, units are often numbered by layer, instead of following a global numbering.

### 4.4 The Learning Process

The memorization of patterns and the subsequent response of the network can be categorized into two paradigms:

- Associative mapping
- Regularity detection

### 4.4.1 Associative mapping

The network learns to produce a particular pattern on the set of input units whenever another particular pattern is applied on the set of input units. The associative mapping can generally be broken down into two mechanisms:

- **Auto-association:** an input pattern is associated with itself and the states of input and output units coincide. This is used to provide pattern completion, i.e to produce a pattern whenever a portion of it or a distorted pattern is presented. In the second case, the network actually stores pairs of patterns building an association between two sets of patterns.
- **Hetero-association:** It is related to two recall mechanisms:

  **Nearest-neighbour:** Here the output pattern produced corresponds to the input pattern stored, which is closest to the pattern presented.

  **Interpolative:** Here the output pattern is a similarity dependent interpolation of the patterns stored corresponding to the pattern presented. This is a variant associative mapping, i.e there is a fixed set of categories into which the input patterns are to be classified.

### 4.4.2 Regularity detection

In regularity detection units learn to respond to particular properties of the input patterns. Whereas in associative mapping the network stores the relationships among patterns, in regularity detection the response of each unit has a particular 'meaning'. This type of learning mechanism is essential for feature discovery and

knowledge representation. Every neural network possesses knowledge which is contained in the values of the connections weights. Modifying the knowledge stored in the network as a function of experience implies a learning rule for changing the values of the weights.

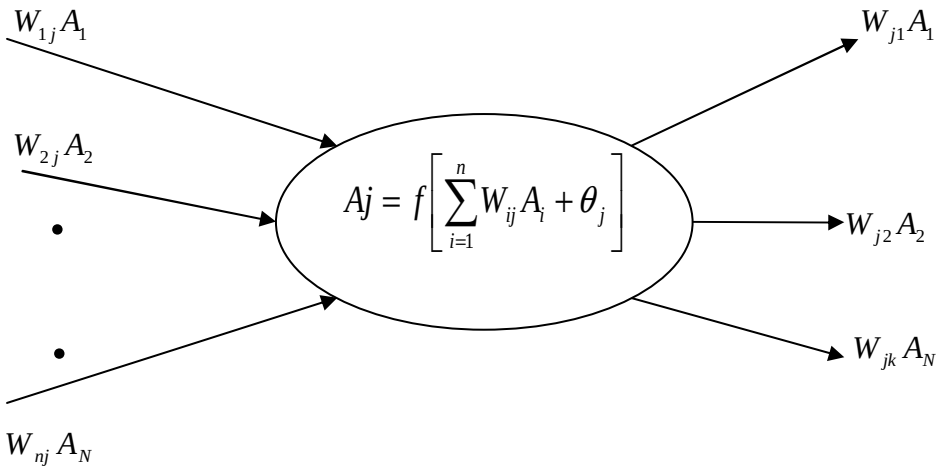



**Figure 10. Weight Matrix**

Information is stored in the weight matrix W of a neural network. Learning is the determination of the weights. Following the way learning is performed, we can distinguish two major categories of neural networks:

- **Fixed networks** in which the weights cannot be changed, ie dW/dt=0. In such networks, the weights are fixed a priori according to the problem to solve.
- **Adaptive networks** which are able to change their weights, ie *d*W/*d*t should not be equal to 0.

  All learning methods used for adaptive neural networks can be classified into two major categories, namely supervised and unsupervised:

**Supervised learning:** It incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be. During the learning process global information may be required [11]. Paradigms of supervised learning include error-correction learning, reinforcement learning and stochastic learning

An important issue concerning supervised learning is the problem of error convergence, ie the minimization of error between the desired and computed unit values. The aim is to determine a set of weights which minimizes the error. One well-known method, which is common to many learning paradigms, is the least mean square (LMS) convergence.

**Unsupervised learning:** Uses no external teacher and is based upon only local information. It is also referred to as self-organization, in the sense that it self-organizes data presented to the network and detects their emergent collective properties. Paradigms of unsupervised learning are Hebbian learning and competitive learning.

We say that a neural network learns off-line if the learning phase and the operation phase are distinct. A neural network learns on-line if it learns and operates at the same time. Usually, supervised learning is performed off-line, whereas unsupervised learning is performed on-line [12].

### 4.5 Transfer Function

The behaviour of an ANN (Artificial Neural Network) depends on both the weights and the input-output function (transfer function) that is specified for the units. This function typically falls into three categories:

- Linear (or ramp)
- Threshold
- Sigmoid

For linear units, the output activity is proportional to the total weighted output. For threshold units, the output are set at one of two levels, depending on whether the total input is greater than or less than some threshold value. For sigmoid units, the output varies continuously but not linearly as the input changes [2]-[12]. Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units, but all three must be considered rough approximations.

To make a neural network that performs some specific task, we must choose how the units are connected to one another and we must set the weights on the connections appropriately. The connections determine whether it is possible for one unit to influence another. The weights specify the strength of the influence.

We can teach a three-layer network to perform a particular task by using the following procedure:

- We present the network with training examples, which consist of a pattern of activities for the input units together with the desired pattern of activities for the output units.
- We determine how closely the actual output of the network matches the desired output.
- We change the weight of each connection so that the network produces a better approximation of the desired output.

## 4.6 Training algorithms for Neural Networks

The Neural Network has to be configured before it can be used for applications. This configuration of neural network is called as training, in which the parameters of the network are adjusted to the optimum values, such that the network exhibits the desired properties [11]. The training required that the network parameters follow an updated rule, which is called as training algorithm.

Based on the way weights are updated, training is classified in two ways:

**Online or Pattern-wise training:** In this mode of training the weights are updated for each error. Starting from the first input instance of the data-set, the error for each input is calculated as shown in the above equation. The amount weight can be given by

$$\Delta w = -\eta \frac{\partial \varepsilon}{\partial w} \tag{4.2}$$

Where $\eta$ is the learning rate? The procedure is repeated until the last instance of the data-set.

**Batch or epoch wise training:** In this mode the weights are updated on the calculation of the total error $\mathcal{E}_{Total}$ the weights are updated when a complete batch or data-set are presented to the network. The amount of weight change is given by

$$\Delta w = -\eta \frac{\partial \mathcal{E}_{Total}}{\partial w} \tag{4.3}$$

### 4.6.1 Back propagation algorithm

The backpropagation algorithm is a supervised learning method for multi-layered feedforward neural networks using sigmoidal activation functions. It was developed by Paul Werbosin in 1974 and was later extended by Rumelhart, Hinton and Williams in 1986 this was the first network with more than one hidden layer. It is a gradient descent local optimization technique, it involves backward error correction of the network weights [28]-[36]. For non-linear applications the backpropagation algorithm has a local minima problem, it cannot find the global minima.

**Architecture of the Network**

The Backpropagation architecture consists of an input layer, a minimum of one hidden layer and an output layer. The nodes in each layer are fully connected to the nodes in previous and next layers. Each connection is associated with a synaptic weight.

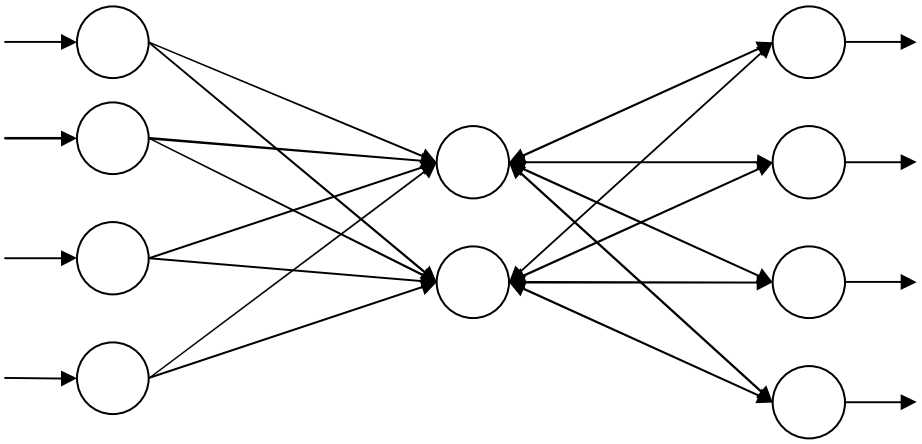



**Figure 11. Backpropagation architecture**

The flow through the network can be described as follows:

- Input to Hidden layer: The input layer loads data from the input vector X, and sends them to the first hidden layer.
- Hidden layer: The hidden layer units receive weighted input and transfer them to the next hidden or output layer using one of the transfer functions (sigmoid).
- As the information propagates through the network all the summed inputs and output states are computed in each processing unit.
- Backpropagation from the output to the hidden layers: the scaled local error and weighted increments or decrements are computed for each layer backwards, starting from the output layer and ending at the first hidden layer, and finally weights are updated this process is repeated until the error is minimized .

**Computation involved in the Network:**

Let us consider that the input, hidden and the output layer consists of N, K and M Neurons respectively. Let us take the output of the m-th output node due to p-th input pattern is given by $O_{pm}$, the output of the k-th hidden node for the p-th input pattern is given by $\bar{O}_{pk}$ the biases $\bar{\theta}_k$ and $\theta_m$ are associated with the k-th hidden node and the m-th ouput node respectively [28]-[36]. Let $\omega_{km}$ be the weight between the m-th output neuron and the k-the hidden neuron and $\bar{\omega}_{nk}$ be the weight between k-th hidden neuron and n-th input neuron. The desired output for the m-th output neuron due to p-th input pattern is given by $\tau_{pm}$. The input for the n-th input neuron due to p-th input pattern is denoted by $x_{pn}$ (where $x_{pn}$ is either 0 or 1). Using this definition the output of the k-th node in the hidden layer is given by:

$$\bar{O}_{pk} = f\left(\sum_{n=1}^{N} \bar{\omega}_{nk}\, x_{pn} + \bar{\theta}_k\right) \tag{4.4}$$

Where $f$ is the activation function (sigmoid) defined as

$$f(x) = 1/1 + e^{-x} \tag{4.5}$$

Similarly the output of the m-th node in the output layer is given by:

$$O_{pm} = f\left(\sum_{k=1}^{K} \omega_{km}\, \bar{O}_{pk} + \theta_m\right) \tag{4.6}$$

We define sum of the squared error of the system to be:

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M}\left(\tau_{pm} - O_{pm}\right)^2 \tag{4.7}$$

The backpropagation learning algorithm is to change the current weights $\omega_{km}$ and $\bar{\omega}_{nk}$ iteratively such that the system error function E is minimized. The weight updates are proportional to the partial derivative of E with respect to $\omega_{km}$.

$$\frac{\partial E}{\partial \omega_{km}} = \frac{\partial E}{\partial O_{pm}} \cdot \frac{\partial O_{pm}}{\partial E} \tag{4.8}$$

$$\text{Where} \quad \frac{\partial E}{\partial O_{pm}} = O_{pm} - \tau_{pm} \quad \text{and} \quad \frac{\partial O_{pm}}{\partial E} = O_{pm}\left(1 - O_{pm}\right)\bar{O}_{pk} \; ; \tag{4.9}$$

And the partial derivative of E with respect to $\bar{\omega}_{nk}$ is:

$$\frac{\partial E}{\partial \bar{\omega}_{nk}} = \sum_{m=1}^{M} \frac{\partial E}{\partial O_{pm}} \cdot \frac{\partial O_{pm}}{\partial \bar{O}_{pk}} \cdot \frac{\partial \bar{O}_{pk}}{\partial \bar{\omega}_{nk}} \tag{4.10}$$

where

$$\frac{\partial O_{pm}}{\partial \bar{O}_{pk}} = O_{pm}\left(1 - O_{pm}\right)\omega_{km} \quad \text{and} \quad \frac{\partial \bar{O}_{pk}}{\partial \bar{\omega}_{nk}} = \bar{O}_{pk}\left(1 - \bar{O}_{pk}\right)x_{pn} \tag{4.11}$$

The weight change for the (n+1)-th iteration can be expressed as follows (where $\eta$ and $\alpha$ are the learning rate and the momentum of the gradient method respectively).

$$\Delta\omega_{km}(n+1) = -\eta\sum_{p=1}^{P}\left(\frac{\partial E}{\partial \omega_{km}}\right) + \alpha\omega_{km}(n) \tag{4.12}$$

$$\Delta\bar{\omega}_{nk}(n+1) = -\eta\sum_{p=1}^{P}\left(\frac{\partial E}{\partial \bar{\omega}_{nk}}\right) + \alpha\bar{\omega}_{nk}(n) \tag{4.13}$$

or

$$\Delta\omega_{km}(n+1) = \eta\sum_{p=1}^{P}\delta_{pm}\bar{O}_{pk} + \alpha\Delta\omega_{km}(n) \tag{4.14}$$

where

$$\delta_{pm} = (\tau_{pm} - O_{pm})O_{pm}(1-O_{pm}) \tag{4.15}$$

$$\Delta\bar{\omega}_{nk}(n+1) = \eta\sum_{p=1}^{P}\bar{\delta}_{pk}\, x_{pn} + \alpha\Delta\bar{\omega}_{nk}(n) \tag{4.16}$$

where

$$\bar{\delta}_{pk} = \bar{O}_{pk}(1-\bar{O}_{pk})\sum_{m=1}^{M}\delta_{pm}\omega_{km} \tag{4.17}$$

The biases $\theta_m$ and $\bar{\theta}_k$ are update similar to $\omega_{km}$ and $\bar{\omega}_{nk}$ using equations (4.12)-(4.14).

# CHAPTER 5

# IMAGE/VIDEO COMPRESSION USING NEURAL NETWORKS

Apart from the existing technology on image compression represented by series of JPEG, MPEG and H.26x standards, new technology such as neural networks and genetic algorithms are being developed to explore the future of image coding. The various architectures of neural networks discussed in the previous chapters can be used for the compression of still images and motion pictures. Research on neural networks of image compression is still making steady advances which could have a tremendous impact upon the development of new technologies and algorithms in this subject area [2]-[12]. Successful applications of neural networks to vector quantization have now become well established, and other aspects of neural network involvement in this area are stepping up to play significant roles in assisting with traditional technologies.

## 5.1 Back-propagation image compression.

### 5.1.1 Back propagation Neural Network.

Back-propagation neural networks can be directly applied to image compression coding. The neural network structure can be illustrated as three layers, one input layer, one output layer and one hidden layer. The input layer and output layer are fully connected to the hidden layer. Compression is achieved by designing the value of *K*, the number of neurons at the hidden layer, less than that of neurons at both input and the output layers.

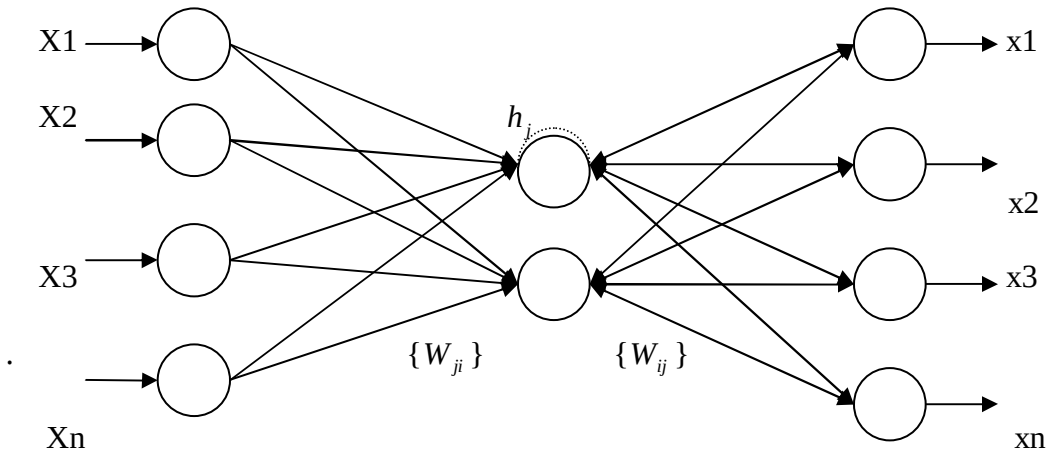



**Figure 12**. **Back-propagation Neural Network**

The input layer and output layer are fully connected to the hidden layer. Compression is achieved by designing the value of *K* which is the number of neurons at the hidden layer which must be less than that of neurons at both input and the output layers. The input image is split up into blocks or vectors of 8×8, 4×4 or 16×16 pixels [8],[9]. When the input vector is referred to as *N*-dimensional which is equal to the number of pixels included in each block, all the coupling weights connected to each neuron at the hidden layer can be represented by $\{W_{ji}, j = 1, 2,\ldots,K$ and $i = 1, 2,..,$ N, which can also be described by a matrix of order *K*x*N*. From the hidden layer to the output layer, the connections can be represented by $\{W_{ij} : 1 \le i \le N, 1 \le j \le K\}$ which is another weight matrix of order $N \times K$. Image compression is achieved by training the network in such a way that the coupling weights$\{Wji\}$ scale the input vector of *N*-dimension into a narrow channel of *K*-dimension ($K<N$) at the hidden layer and producing the optimum output value which makes the quadratic error

between input and output minimum. In accordance with the neural network structure shown, the operation of a linear network can be described as follows:

$$h_j = \sum_{i=1}^{N} W_{ji} x_i \qquad 1 \le j \le K \text{ (For encoding)} \tag{5.1}$$

$$x_i = \sum_{j=1}^{\bar{K}} W_{ij}^{'} h_j \qquad 1 \le i \le N \text{ (For decoding)} \tag{5.2}$$

Where $x_i \in [0,1]$ which means they are the normalized values for the grey scale images with grey levels [0,255]. The reason for normalizing pixel values is neural networks can operate more efficiently when their input and output values are limited to a range of [0, 1]. The above linear network can be transmitted into a nonlinear one by adding a transfer function like sigmoid to the hidden layer and the output layer.

The back-propagation neural network compression is conducted in two phases training and encoding. In the first phase, a set of image samples are fed to train the network using the back-propagation learning rule which uses each input vector as the desired output. This is equivalent to compressing the input into the narrow channel represented by the hidden layer and then reconstructing the input from the hidden to the output layer. The second phase involves the entropy coding of the state vector *hj* at the hidden layer. In the case of adaptive training the entropy coding of these coupling weights is required in order to catch up with some input characteristics that are not encountered at the training stage. The entropy coding is designed as the fixed length binary coding although many advanced variable length entropy coding algorithms are available. One of the reasons for this is the research community is concerned with the part played by neural networks. Therefore, the compression performance can be assessed in terms of the compression ratio or bit rate [10], [11].

For the back propagation narrow channel compression neural network, the bit rate can be defined as follows:

$$\text{bit rate} = \frac{nKT + NKt}{nN} bits / pixel \tag{5.3}$$

where input images are divided into *n* blocks of *N* pixels or *n N*-dimensional vectors; *T* and *t* stand for the number of bits used to encode each hidden neuron output and each coupling weight from the hidden layer to the output layer. When the coupling weights are maintained the same throughout the compression process after training is completed, the term *NKt* can be ignored and the bit rate becomes *KT*/*N* bits/pixel. Since the hidden neuron output is real valued, quantization is required for fixed length entropy coding which is normally designed as 32 level uniform quantization corresponding to 5 bit entropy coding.

This neural network development is in the direction of K-L transform technology which actually provides the optimum solution for all linear narrow channel type of image compression neural networks [3]. When above equations are represented in matrix form, we have

$$[h] = [W]^T [x] \qquad \text{(For encoding)} \tag{5.4}$$

$$[\bar{x}] = [W'][h] = [W'][W]^T [x] \qquad \text{(For decoding)} \tag{5.5}$$

The K-L transform maps input images into a new vector space where all the coefficients in the new space are de-correlated. This means that the covariance matrix of the new vectors is a diagonal matrix whose elements along the diagonal are eigenvalues of the covariance matrix of the original input vectors. Let $e_i$ and $\lambda_i$, *i*=1, 2.. *n*, be eigenvectors and eigenvalues of $c_x$, the covariance matrix for input vector ***x***,

and those corresponding eigenvalues are arranged in a descending order so that $\lambda_i \geq \lambda_{i+1}$, for *i*=1, 2.. *n*. To extract the principal components, *K* eigenvectors corresponding to the *K* largest eigenvalues in $c_x$ are normally used to construct the K-L transform matrix, [*AK*], in which all rows are formed by the eigenvectors of $c_x$. In addition, all eigenvectors in [*AK*] are ordered in such a way that the first row of [*AK*] is the eigenvector corresponding to the largest eigenvalue, and the last row is the eigenvector corresponding to the smallest eigenvalue [4],[5]. Hence, the forward K-L transform or encoding can be defined as:

$$[y][A_K]([x]-[m_x]) \tag{5.6}$$

and the inverse K-L transform or decoding can be defined as:

$$\left[\bar{x}\right] = [A_K]^T[y] + [m_x] \tag{5.7}$$

where $[m_x]$ is the mean value of [*x*] and $[\bar{x}]$ represents the reconstructed vectors or image blocks. Thus the mean square error between ***x*** and $\bar{x}$ is given by the following equation:

$$e_{ms} = E\{(x-\bar{x})^2\} = \frac{1}{M}\sum_{k=1}^{M}(x_k - \bar{x}_k)^2 = \sum_{j=1}^{n}\lambda_j - \sum_{j=1}^{k}\lambda_j = \sum_{j=k+1}^{n}\lambda_j \tag{5.8}$$

where the statistical mean value $E\{.\}$ is approximated by the average value over all the input vector samples which, in image coding are all the nonoverlapping blocks of 4×4 or 8×8 pixels. Therefore, by selecting the *K* eigenvectors associated with the largest eigenvalues to run the K-L transform over input image pixels, the resulting errors between the reconstructed image and the original one can be

minimized due to the fact that the values of $\lambda's$ decrease monotonically. From the comparison between the equation pair (2.4) and (2.5) and the equation pair (2.6) and (2.7), it can be concluded that the linear neural network reaches the optimum solution whenever the following condition is satisfied:

$$[W^{'}][W]^{T} = [A_K]^{T}[A_K] \tag{5.9}$$

Under this circumstance, the neuron weights from input to hidden and from hidden to output can be described respectively as follows:

$$\left[W^{'}\right] = [A_K][U]^{-1}, \tag{5.10}$$

$$[W]^{T} = [U][A_K]^{T} \tag{5.11}$$

where [U] is an arbitrary *K*x*K* matrix and$[U]\left[U\right]^{-1}$ gives an identity matrix of *K*x*K*. Hence, it can be seen that the linear neural network can achieve the same compression performance as that of K-L transform without necessarily obtaining its weight matrices being equal to $\left[A_K\right]^{T}$ and [*AK*].

**5.2 Simulation**

After training the network using one or more frames, we apply the performance phase, which is here equivalent to the coding/decoding process. The hidden layer weight matrix is multiplied by the output of the pre-processor. Then, the bias is added and the output layer transfer function is applied to the result. This result is the output of the hidden layer. The process is repeated to obtain the output of the output layer with the input being the output of the hidden layer.

### 5.3 Post-processing

During decoding, the images are reconstructed using the coding product associated with the input patterns, which will be the output of the hidden layer together with the weights. The reconstructed image will be an approximation of the original one in the decoding phase.

### 5.4 Proposed Image Compression Architecture.

The proposed architecture employs an image/video compression method which uses neural networks in combination with simple motion detection techniques to give an overall improved performance. In general, the network is initially, trained with some frame until the weights are adapted. The adapted weights are used for coding the frame sequence. Since the adapted weights may not be optimal for the particular frame sequence we may need to train the network using frames at regular intervals and code the subsequent frames using the updated netrwork. The detailed description of the architecture is discussed in the following sections.

The second scheme deals with the motion detection techniques. Here, the initial frame, say Frame1, is transmitted through the neural network to the receiving end, while the subsequent frames are coded as follows: Each 8×8 block is compared with the 8×8 block of the previous frame, i.e the 8×8 blocks of Frame2 are compared with Frame1. A bit is used to inform about the existence or not of motion. The blocks for which motion is detected are transmitted through the neural network to the receiving end along with 1 bit. The blocks for which motion has not been detected

remains the same as in the previous frame, which increases the compression ratio without significantly affecting the frame quality.

### 5.4.1 Encoding

The encoding and decoding phases are explained in terms of an example. Consider the video sequence of "hotel" containing a set of 98 frames. The initial frame which will be the first input is divided into an array of 8×8 blocks. Those blocks are given as input to train the neural network architecture until the weights are adapted. We have trained for 100,200,300,400 and 500 epochs. Then, the adapted weights of this initial frame are used for the direct coding of subsequent frames. Thus, the compression is achieved at the hidden layer depending on the number of neurons in the layer and the number of quantization levels used for weights and hidden layer outputs.

### 5.4.2 Decoding process

The compressed data in the hidden layer is passed to the output layer for reconstruction of the images. Therefore, the compressed data for all the frames starting from frame 1 to the last frame is passed to the output layer for reconstruction. The error for each frame is calculated by comparing the reconstructed with the original image. These error values are used for the calculation of the signal to noise ratio of the images for particular compression ratios.

The advantage of the above method is that the training is not done often which increases the technique's processing speed while maintaining the compression ratio.

**Transmitting the motion detected between frames.**

In this scheme, we train the network using the initial frame (F1) until the weights are adapted, the adapted weights are transmitted for direct coding of F1. Now, at the transmitting end, frame2 (F2) is split into 8×8 blocks. In our case, since the images are of size 512×480 we get 3840 8×8 blocks for each frame. Therefore, the 8×8 blocks in frame2 (F2) are compared with the 8×8 blocks of frame1 (F1) and checked for motion based on the following equation:

$$\text{M.D} = \text{abs}\left|\sqrt{\sum_{x}\sum_{y}\left(F1_{m,n}.(X,Y) - F2_{m,n}.(X,Y)\right)^2}\right| \qquad (5.12)$$

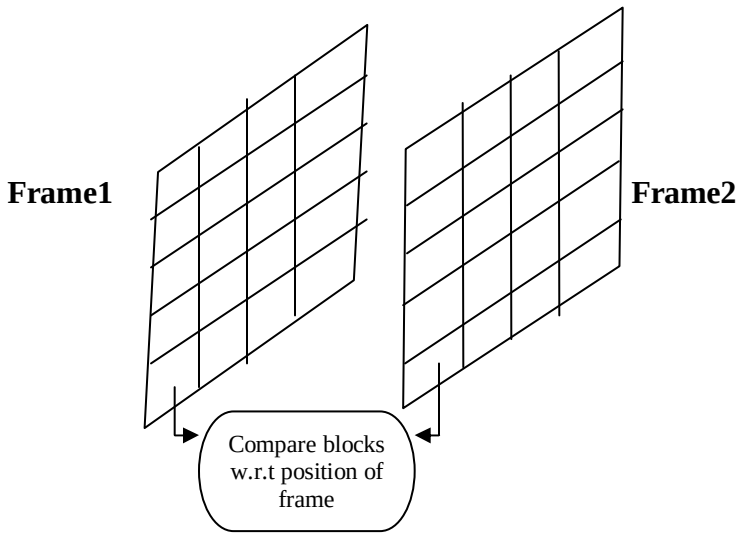



**Figure 13. Motion Detection**

The information about the detected 8×8 blocks is stored in an array which is sent to the receiving end. Thus, after we complete the comparison of all blocks, we

transmit the 8×8 blocks of frame2 (F2) where motion is detected through the neural network decoding part at the receiver which has already received the adapted weights at the receiving end, these blocks are reordered in their original position to construct frame2 (F2). The same process is carried out for subsequent frames (i.e. frame2 (F2) is compared with frame3 (F3)) till the last frame of the video sequence.

This technique has the advantage of transmitting only the motion part in combination which gives an additional compression compared to the case where all blocks are transmitted. This technique is helpful for motion pictures where the change between frames is relatively small.

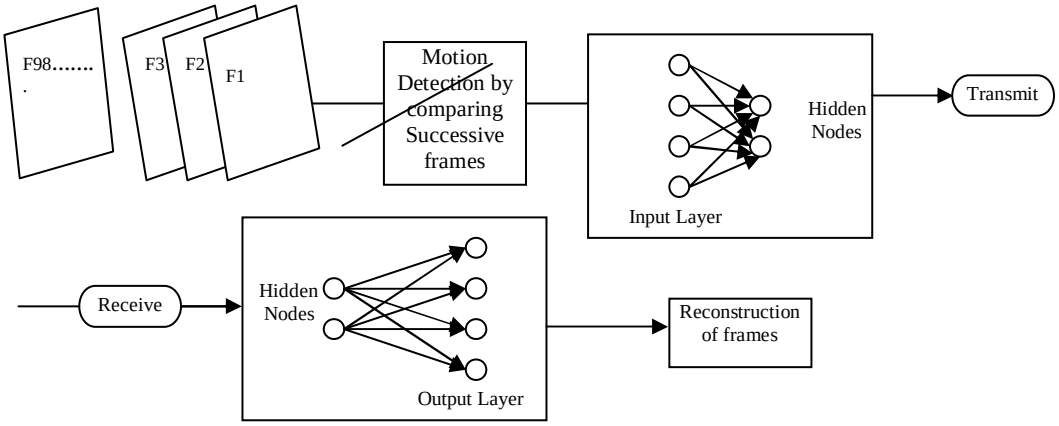



**Figure 14. Flow of the proposed scheme**

**Retraining Frames at regular intervals.**

In this case, we train the network using frames of the motion picture at regular intervals. Initially, frame1 is used to train the network and obtain the first set of weights (for 100,200 or 300 epochs). The adapted weights are used for coding of the trained frame and the subsequent frames until a new weight update takes place. In our case, we consider the training frequency to be four. For instance, after the first weight update, the weights are again updated using the fifth frame. Then, the new weights are used to code the next four frames starting from frame5. As the training frequency decreases, the compression ratio increases and vice-versa.

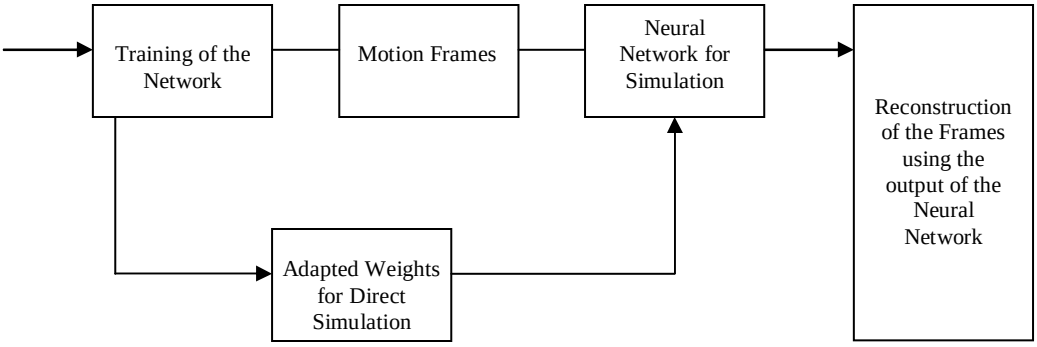



**Figure 15. Retraining frames**

**Self-Adaptive Training:**

This is a modification of the above scheme in which, instead of training the frames at regular intervals, we train the frames based on a threshold value. In this case, frame1 is trained initially and then the following frames are coded using the obtained weights. The same set of weights is used until an error-based threshold value is reached. The threshold value is calculated based on the mean square error of the reconstructed frame with respect to the original one. Once the threshold is reached then the next frame in the series is used to train the network in order to obtain a new set of weights. The updated weights are used for coding the subsequent frames. Based on this approach, training is performed only when the quality of the reconstructed frames is degraded significantly. This technique results in higher compression ratios compared to the technique in which retraining is performed at regular intervals.

**Proposed Technique:** Here, motion is used in combination with retraining to improve the compression ratio. The procedure followed here is similar to the motion detection one. However, similarly to the self-adaptive training technique, when the error for a frame exceeds a certain threshold value, retraining is performed to update the weights. The updated weights help reducing the error for future frames, which then results in transmitting a smaller number of blocks. This in-turn increases the compression ratio. The proposed scheme helps in drawing some useful conclusions with respect to compression ratio and signal-to-noise ratio.

# CHAPTER 6

# RESULTS

The video compression techniques presented in chapter 5 are tested and results are presented for various scenarios using a set of 98 frames of a "hotel" motion picture. The comparisons are made based on the signal-to-noise ratio vs compression ratio.

## 6.1 Comparison of results for various test scenarios.

Image/Video compression results are presented for various test scenarios with the help of a motion picture containing a set of 98 frames. The set of frames are tested for motion detection, the retraining frames method and the self-adaptive method. The results obtained from the above tests were useful in drawing some conclusions regarding the aforementioned techniques. The compression ratio and peak-signal-to-noise ratios (PSNR) are calculated based on the following formulas for all the test scenarios.

$$\textbf{Compression ratio} = \frac{\left[K \times (L \times M + N) + (A \times T \times R) + (B \times W \times W1)\right]}{T \times P \times Q} \qquad (6.1)$$

where

K = No. of blocks transmitted.
L = No. of outputs from hidden layer
M = bits per output
N = No. of bits for mean
T = Total no. of blocks
P = No. of pixels per block
Q = No. of bits per pixel
R = 1bit per block to send the motion information.
W = No. of weights
W1 = No. of bits per weight
A = 1, if motion is detected
A = 0, if motion is absent
B = 1, if retraining is done
B = 0, if retraining is not done

$$\mathbf{PSNR} = 10 \times \log_{10}(1/error) \tag{6.2}$$

**Case1:** Initially, the "Lena" image is trained for different epochs (100,300,500) and 4 hidden nodes in the network. The network is also tested for a still image with the above parameters. We can see that as the training was increased to 500 epochs, the weights seem to be better adapted to the particular image, and thus the quality of the reconstructed image is higher compared to the one trained for 100 or 300 epochs. Nevertheless, training for 500 epochs has higher processing requirements compared to the other two cases. Moreover, Figure 16 illustrates that as the compression ratio increases; the difference in terms of PSNR between the three different cases becomes negligible.

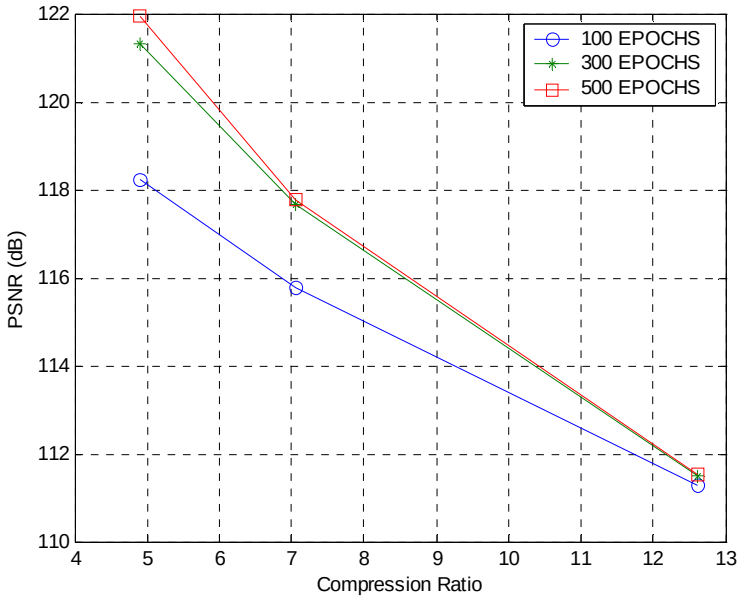



**Figure 16. Performance of the Lena Image**

**Case2:** Initially, we train the network using the initial frame of a motion picture containing a set of 98 frames of a hotel sequence. The trained weights, using the initial frame, are used for the direct simulation of all the 98 frames. From Figure 17, we can conclude that the video picture quality is higher for as the number of training epochs increases. However, as in the example of Figure 16, it can be seen that the difference in terms of PSNR decreases as the compression ratio decreases.

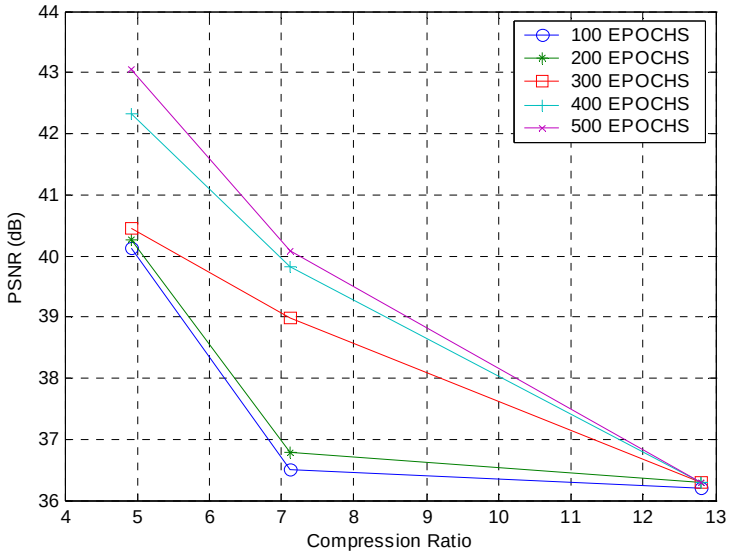



**Figure 17. Direct Simulation of Frames.**

**Case3:** Here two motion pictures are concatenated. These are the "hotel" sequence which contains relatively complex images, and the "golf" sequence which contains simple images. The initial frame of the hotel sequence is trained and these weights are used to code the mixed video sequence i.e. frame1 to frame 98 for hotel sequence and frame 99 to frame 149 for golf sequence. We can see from Figure 18 that there is a sudden change in the PSNR at frame 99 because of the transition from hotel image to golf image. In this case, the PSNR increased. Since the golf image is a simple image

and the information contained in it is most probably included in the hotel sequence, the network trained using the hotel sequence will be capable of producing a high quality reconstruction of the frames. On the other hand, when the same experiment is repeated by placing the golf sequence prior to the hotel sequence, there is a sudden PSNR change at frame 51 which is the frame at which the transition from the simple (golf) to the complex (hotel) image sequence occurs. This shows that the network using the weights obtained after training the first frame of the golf sequence is not capable of successfully coding the hotel sequence which contains more significant information.

In addition to the above observations, it is important to mention that the PSNR for the golf sequence, when the network is trained using the first frame of the hotel sequence, is higher than the PSNR for the golf sequence when the network is trained using the first frame of the golf sequence. This may be surprising at first, however it should be expected. Since the hotel sequence provides a “better” set of blocks for training the network, all frames of the golf sequence can be effectively coded. However, when the first frame of the golf sequence is used to generate the network’s weights, the subsequent frames of the golf sequence can not be successfully represented by the information included in the network weights. This happens because this information is provided by the “not so good” set of blocks of the first frame of the golf sequence.

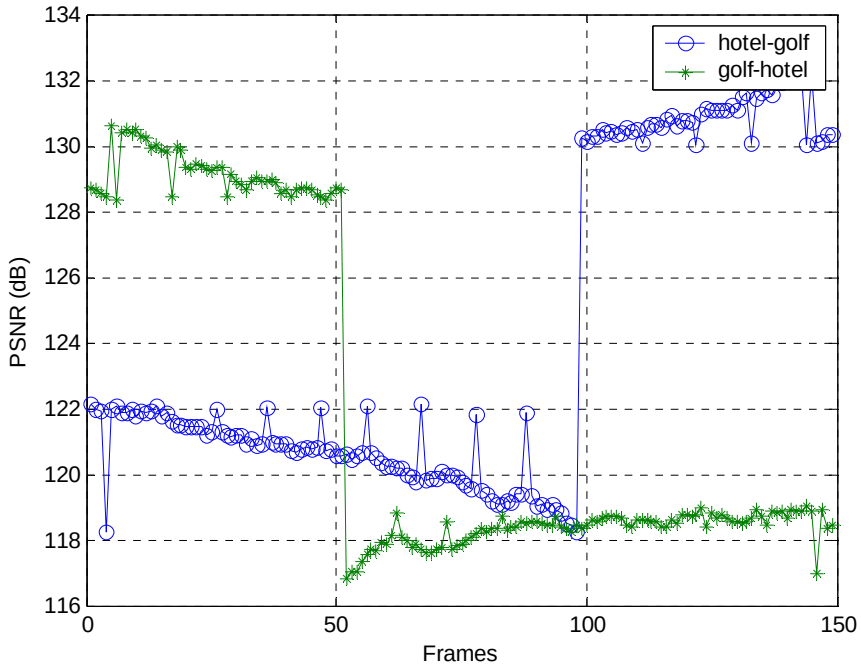



**Figure 18. Comparison of Hotel-golf/golf-hotel sequences**

**Case4:** Here retraining of video frames is done at regular intervals (3, 4, 5, 6 frames) to update the weights of the neural network for improving the quality of the video sequence, since the initial set of weights may not be "good" for coding the frames at a later stage. From the Figure 19, we can see that as the retraining frequency increases, the quality of the reconstructed frame sequence increases, however the compression ratio is decreased and more processing is required.

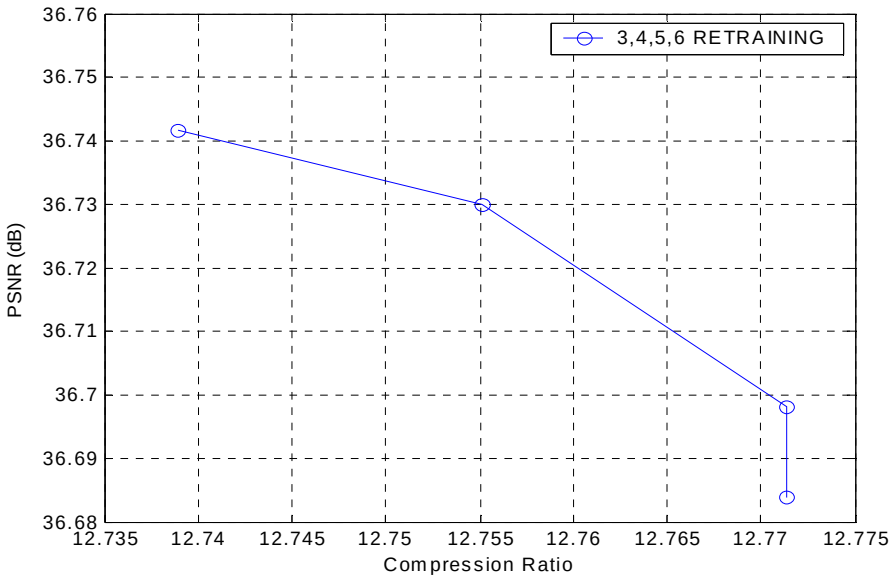



**Figure 19. Retraining at regular intervals**

**Case5:** Here, retraining is done at regular intervals of 10 frames and the updated weights are used in between the intervals (Example 15$^{th}$ frame). This technique is useful when we have parallel processors where training takes place continuously and the weights are updated while the coding takes place in parallel.

| | | |
|---|---|---|
| **C.R(4-nodes)** | 12.7826 | 12.7878 |
| **PSNR** | 36.6264 | 36.5285 |
| **C.R(8-nodes)** | 7.1057 | 7.1074 |
| **PSNR** | 39.9610 | 39.3818 |
| **C.R(12-nodes)** | 4.9205 | 4.9213 |
| **PSNR** | 41.1565 | 40.7913 |

**Table 3: Retraining for different nodes**

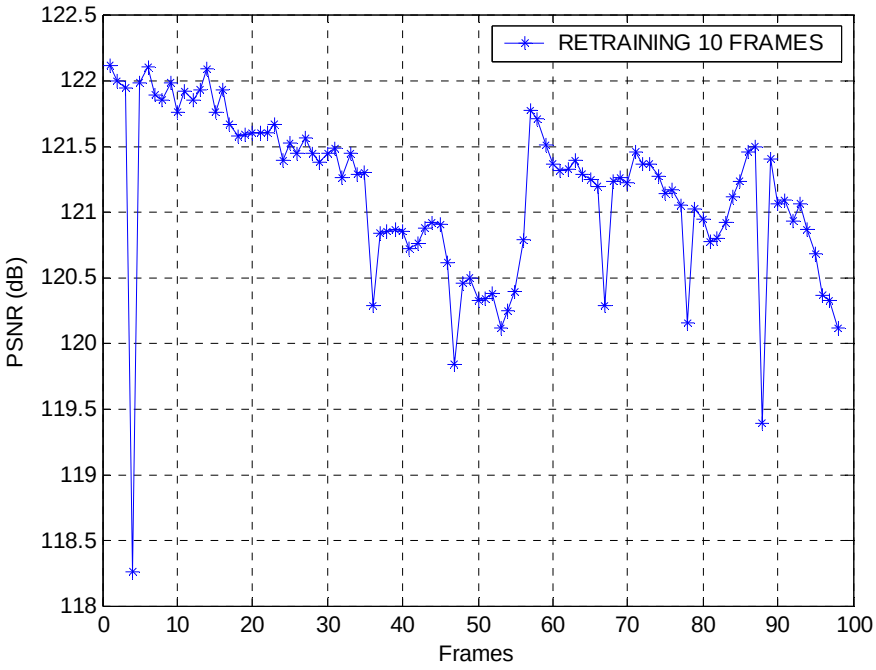



**Figure 20. Retraining every 10th frame**

**Case6:** In this case there is a comparison between the direct coding and retraining techniques. Figure 21 indicates that retraining using frames at regular intervals helps in maintaining the quality of the video sequence with some additional overhead of 10 sets of weights.

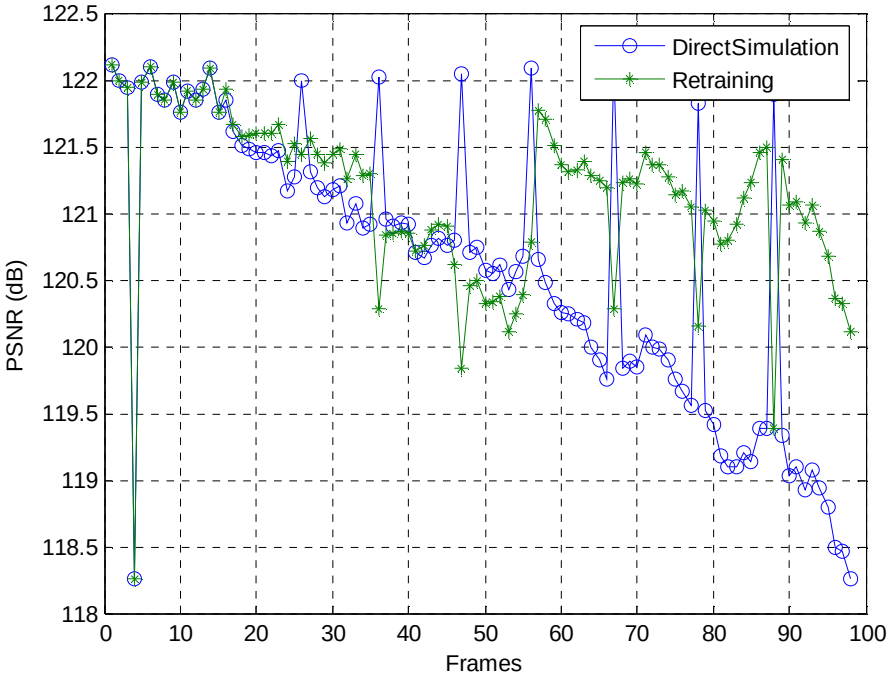



**Figure 21. Comparison between Direct simulation and Retraining**

**Case7:** In this method, retraining is done only when the error of the reconstructed image exceeds certain threshold value. The network automatically retrains when the error exceeds that threshold. This method is useful for reducing the overhead when compared to retraining at regular intervals. The compression ratios are higher compared to training at regular intervals.

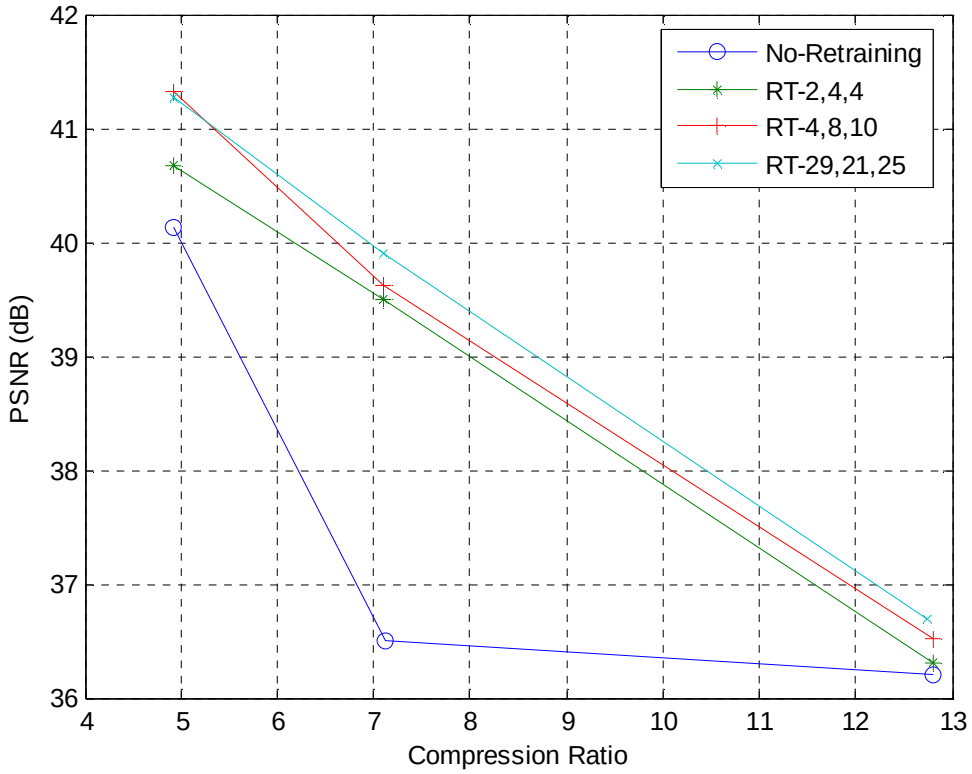



**Figure 22. Self-Adaptive Network**

| | | | |
|---|---|---|---|
| **C.R(4-nodes)** | 12.7965 | 12.7930 | 12.7497 |
| **PSNR** | 36.3063 | 36.5262 | 36.6852 |
| **C.R(8-nodes)** | 7.1090 | 7.1068 | 7.0998 |
| **PSNR** | 39.5036 | 39.6193 | 39.8968 |
| **C.R(12-nodes)** | 4.9220 | 4.9205 | 4.9166 |
| **PSNR** | 40.6794 | 41.3236 | 41.2650 |

**Table 4: Self-adaptive network**

**Case8:** Here, we apply the motion detection technique in which the frames in the sequence are split into 8×8 blocks. These blocks are then compared to the 8×8 blocks in the next frame, and if there is a motion detected that particular block is transmitted through the neural network to the receiving end. The received blocks are placed in their respective positions to construct the new frame. Thus, the frame at the receiving

end is built based on the previous frames blocks and newly coded blocks. Figure 23 shows the results of this approach using 4, 8, and 12 hidden nodes. Furthermore, different thresholds have been used for motion detection for each one of the three cases. In this method, significantly high compression ratios are attained. Nevertheless, as indicated from Figure 23, in certain cases, using a smaller number of hidden nodes for increasing the compression ratio may be preferred over using the motion detection approach. In any case, using the motion detection approach for small motion detection thresholds (which implies that only few blocks will be considered as showing lack of motion) increases the compression ration without affecting the PSNR.

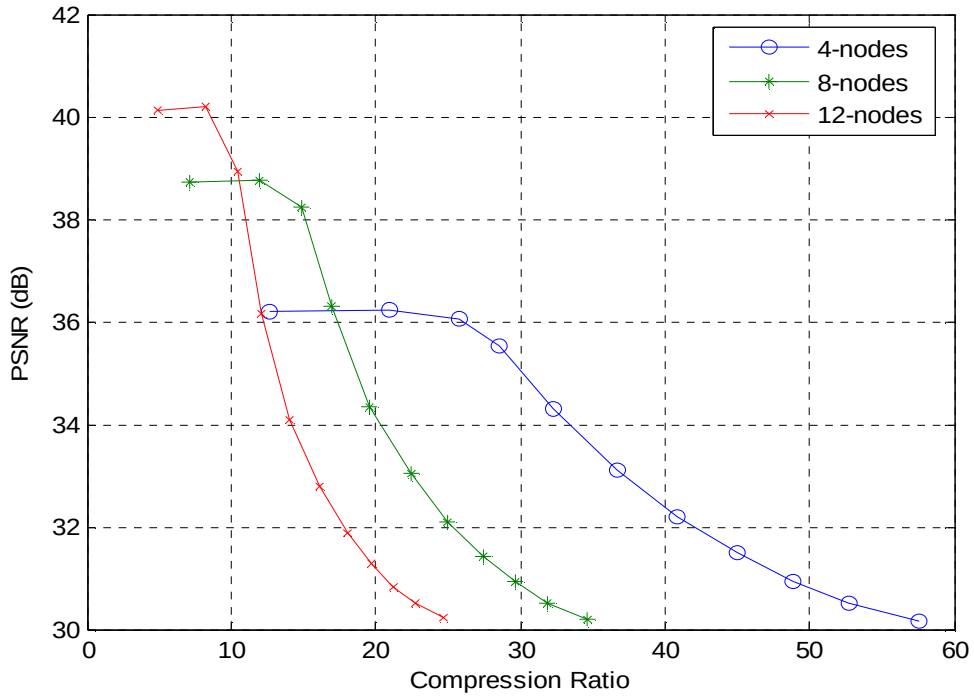



**Figure 23. Motion Detection**

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **C.R 4node** | 12.61 | 20.98 | 25.79 | 28.60 | 32.34 | 36.72 | 40.911 | 45.01 | 48.921 | 52.806 | 57.544 |
| **PSNR** | 36.19 | 36.22 | 36.06 | 35.53 | 34.30 | 33.11 | 32.191 | 31.47 | 30.939 | 30.519 | 30.171 |
| **C.R 8node** | 7.084 | 11.96 | 14.82 | 16.91 | 19.59 | 22.43 | 25.015 | 27.39 | 29.640 | 31.881 | 34.642 |
| **PSNR** | 38.73 | 38.77 | 38.21 | 36.29 | 34.34 | 33.02 | 32.104 | 31.42 | 30.91 | 30.523 | 30.202 |
| **C.R 12nod** | 4.925 | 8.285 | 10.46 | 12.12 | 14.10 | 16.14 | 18.000 | 19.66 | 21.232 | 22.724 | 24.616 |
| **PSNR** | 40.12 | 40.18 | 38.93 | 36.16 | 34.08 | 32.79 | 31.880 | 31.26 | 30.81 | 30.510 | 30.224 |

**Table 5: Motion Detection for different nodes**

**Case9:** This case presents a comparison between the technique that uses motion detection and the one that uses motion with retraining. Figure 24 and Table 6 illustrate that, for a given threshold value, the motion with retraining technique has resulted in higher compression ratios for a given PSNR when compared to the technique that only uses motion detection. This is because retraining updates the weights so that the corresponding error is not allowed to increase considerably. As a result, only few blocks are transmitted to the receiving end, due to motion detection, which in-turn increases the compression ratio.

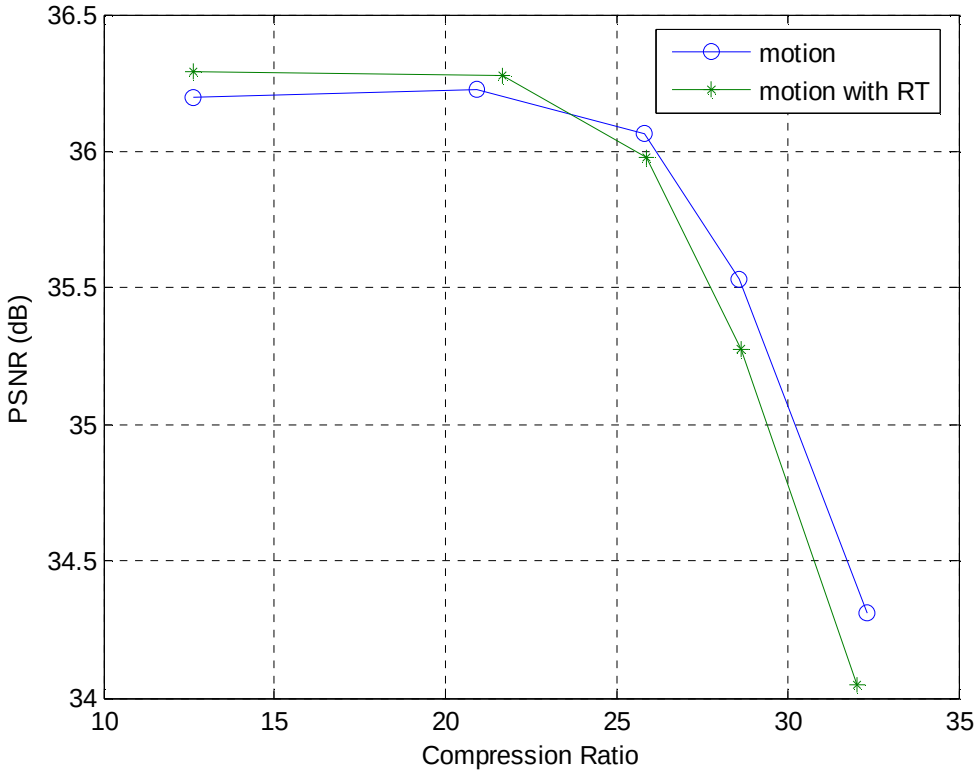



**Figure 24. Combination of motion with retraining.**

| Error | 0.001 | 0.005 | 0.010 | 0.015 | 0.02 |
|---|---|---|---|---|---|
| C.R(motion) | 12.6117 | 20.8914 | 25.7960 | 28.6054 | 32.3459 |
| PSNR | 36.1964 | 36.2278 | 36.0678 | 35.5321 | 34.3099 |
| C.R(motion,RT) | 12.6100 | 21.6539 | 25.8658 | 28.6402 | 32.0302 |
| PSNR | 36.2888 | 36.2778 | 35.9758 | 35.2768 | 34.0465 |

**Table 6: Motion with Retraining**

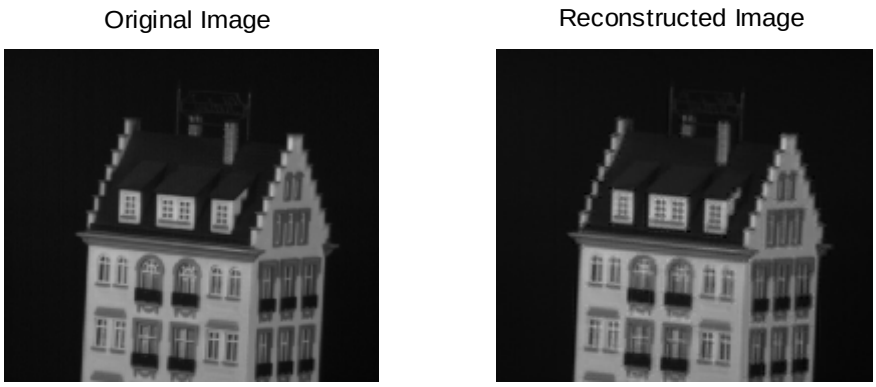



**Figure 25: Comparison of Original and reconstructed Images**

Figure 25 presents a comparison between the original and the reconstructed image that has gone through motion with retraining.

# CHAPTER 7

# DISCUSSIONS AND CONCLUSIONS

In this thesis, we have discussed various video compression schemes. The results were useful in drawing important conclusions about those schemes. The algorithms implemented and tested are mainly based on the idea of neural network based image compression.

Compared to other existing neural network schemes, the major advantage of the proposed technique is that it provides better PSNR for a given compression ratio. In general, it was shown that the combination of the neural network techniques with some basic motion detection helped in achieving higher compression ratios. Neural Network weight retraining improved the image quality compared to previous techniques. Moreover, the Self-adaptive retraining achieved even higher PSNR for a given compression ratio.

Future work includes incorporation of lossless techniques to supplement the neural network approach.

# REFERENCES


[1] Christopher E.Cramer and Erol Gelenbe, "Video Quality and Traffic QoS in Learning-Based Subsampled and receiver-Interpolated Video Sequences" in Proc. SPIE: "Visual Communications and Image processing",vol-18,No.2,Feb 2000.

[2] D. Anthony ,"A comparison of image compression by a neural network and principle component analysis," in Proc.Int.Joint Conf. Neural networks, 1990, pp.339-344.

[3] S.Carrato," Neural Networks for image compression" in Neural Networks: Advances and Applications 2. Amsterdam, the Netherlands: Elsevier, 1992, pp177-198.

[4] S.Carrato and S.Marsi " parallel Structure based on Neural networks for image compression,"Electron lett,vol.28,no.12,pp.1152-1153.

[5] Y.W.Chiang , " Motion estimation using a neural network,", In proc,IEEE Int.symp circuitsand systems,1990,pp.2516-2519.

[6] G.W.Cottrell ,P.Munro and D.Zipser,"Image compression by backpropagation: An example of extensional programming."in models of cognition:AReview of Cognition Science ,N.E.Sharky,Ed Norwood,Ablex,1989.

[7] S.H.Courellis."An artificial Neural Network for motion detection and speed estimation",in proc,Int.Joint Conf.Neural Networks.1990,pp.407-421

[8] W.C.Feng,"Real-Time neuro processor for adaptive image compressionbased upon frequency sensitive learning" in proc, Joint Conf.Neural Networks. 1991,pp.429

[9] E.Gelenbe ,"stability of the random neural network model",neural Comput,vol2,no.2,pp.239-247,1990.

[10] E.Gelenbe and M.Sungur,"Image Compression with the random neural network",presented at the .Int conference.Artificial neural networks, the Netherlands,1994.

[11] R.M .Gray ,"Vector Quantization",IEEE Acoust.,Speech,Signal Processing Mag.,vol1,no.2,pp.4-29.

[12] R.Kohno,"Image compression using a neural network with learning capability of variable function of the neural unit.," in visual communication and image processing:SPIE,1990,pp.69-75.

[13] Sikora, T.,"MPEG digital video-coding standards", Signal Processing Magazine, IEEE Volume 14, Issue 5, Sept. 1997 Page(s):82 - 100

[14] Pao-Chi Chang; Ta-Te Lu,"A scalable video compression technique based on wavelet transform and MPEG coding",Consumer Electronics, IEEE Transactions on Volume 45, Issue 3, Aug. 1999 Page(s):788 - 793

[15] Dapeng Wu; Hou, Y.T.; Wenwu Zhu; Hung-Ju Lee; Tihao Chiang; Ya-Qin Zhang; Chao, H.J.; "On end-to-end architecture for transporting MPEG-4 video over the Internet",Circuits and Systems for Video Technology, IEEE Transactions on Volume 10, Issue 6, Sept. 2000 Page(s):923 - 941

[16] R.Schäfer and T.Sikora, “Digital Video Coding Standards and Their Role in Video Communications,” in *Proceedings of the IEEE*, vol. 83, 1995, pp. 907-923.

[17] L.Chiariglione, “MPEG and Multimedia Communications,” *IEEE Trans. Circuits and Systems for Video Technology*, vol. 7, no. 1, pp. 5-18, Feb. 1997.

[18] T.Sikora, “The MPEG-4 Video Standard Verification Model,” *IEEE Trans. Circuits and Systems for Video Technology*, vol. 7, no. 1, pp. 19-31, Feb. 1997.

[19] T.Sikora, “MPEG-4 Very Low Bit Rate Video,” in *Proc. IEEE ISCAS Conference*, Hong Kong, June 1997.

[20] In, J.; Shirani, S.; Kossentini, F.;”JPEG compliant efficient progressive image coding”,Acoustics, Speech, and Signal Processing, 1998. ICASSP '98. Proceedings of the 1998 IEEE International Conference on Volume 5, 12-15 May 1998 Page(s):2633 - 2636 vol.

[21] Rao, K.R.; Huh, Y. “JPEG 2000”,Video/Image Processing and Multimedia Communications 4th EURASIP-IEEE Region 8 International Symposium on VIPromCom 16-19 June 2002 Page(s):1 – 6

[22] Leu-Shing Lan; Reed, I.S. “An improved JPEG image coder using the adaptive fast approximate Karhunen-Loeve transform (AKLT)” Speech, Image Processing and Neural Networks, 1994. Proceedings, ISSIPNN '94., 1994 International Symposium on 13-16 April 1994 Page(s):160 - 163 vol.1

[23] Wallace, G.K,”The JPEG still picture compression standard”, Consumer Electronics, IEEE Transactions of Volume 38, Issue 1, Feb. 1992 Page(s):xviii - xxxiv

[24] Lakhani, G.” Modified JPEG Huffman coding” ,Image Processing, IEEE Transactions on Volume 12, Issue 2, Feb. 2003 Page(s):159 - 169

[25] S.Wu and A.Gersho, “Rate-constrained picture-adaptive quantization for JPEG baseline coders,” *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 5, pp. 389-392, Apr. 1993.

[26] A.Watson, “DCT quantization matrices visually optimized for individual images,” *Proc. SPIE Human Vision, Visual Processing, and Digital Display IV*, 1993.

[27] M.Orchard and K.Ramchandran, “An investigation of wavelet-based image coding using an entropy-constrained framework,” *Proc. IEEE Data Compression Conf.*, pp. 341-350, Mar. 1994.

[28] www.ph.tn.tudelft.nl/Courses/FIP/frames/fip.html

[29] www.faqs.org/faqs/jpeg-faq/part1/

[30] www.debugmode.com/imagecmp/

[31] www.acm.org/crossroads/xrds6-3/sahaimgcoding.html

[32] www.faqs.org/faqs/mpeg-faq/part1/

[33] www.autosophy.com/videcomp.htm

[34] www.ee.bgu.ac.il/~greg/graphics/compress.html

[35] pascalzone.amirmelamed.co.il/Graphics/JPEG/JPEG.htm

[36] telin.rug.ac.be/~philips/elis/philips/imagecompression.shtml

# APPENDIX

# MATLAB CODES

Functions used for Single-structure Neural Network compression:

- **mynewff**-Creates a feedforward Neural Network
- **mysim**-Simulates the Network and returns the output.

Functions used for proposed architecture:

- **directsim**-Used for the direct simulation of frames.
- **retrain**-Used for retraining of frames at regular intervals.
- **motion**-Used for motion detection between frames.

Functions both common to Neural Network and proposed architecture:

- **Image_to_blocks**-Breaks up the image of UxV size into zxz blocks and changes the Dimensions.
- **reconstruct-** Performs inverse of image_to_block operation.

Script which executes all the three cases.

- **Compute.**

Test Images

Lena.tiff and a set of 98 frames of a motion picture named hotel.seq1….98

# Vita

Prem Kovvuri was born in Tanuku, INDIA. He graduated from Helapuri Junior College, as Juniorate in 1996 .In Fall 1997 he began studies in Electrical/Electronics Engineering at Osmania University,Hyderabad and graduated with a Bachelor of Engineering degree in August 2001.In the fall of 2001 he came to the University of New Orleans to pursue graduate studies in the Electrical Engineering Department. He worked as a Teaching Assistant in the Department of Electrical Engineering.