UNIVERSITY OF MIAMI

EDGE PRESERVING IMAGE COMPRESSION FOR MAGNETIC RESONANCE IMAGES USING DANN-BASED NEURAL NETWORKS

By

Tat Chee Wan

A THESIS

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Master of Science in Electrical and Computer Engineering

Coral Gables, Florida

August, 1993

UNIVERSITY OF MIAMI

A thesis submitted in partial fulfillment of

the requirements for the degree of

Master of Science in Electrical and Computer Engineering

EDGE PRESERVING IMAGE COMPRESSION FOR MAGNETIC RESONANCE IMAGES USING DANN-BASED NEURAL NETWORKS

Tat Chee Wan

Approved:

Wan, Tat Chee (M.S., Electrical and Computer Engineering)

EDGE PRESERVING IMAGE COMPRESSION FOR MAGNETIC RESONANCE IMAGES USING DANN-BASED NEURAL NETWORKS. (August, 1993)

Abstract of a master’s thesis at the University of Miami.

Thesis supervised by Professor Mansur R. Kabuka.

No. of pages in text: 51.

With the tremendous growth in imaging applications and the development of filmless radiology, the need for compression techniques which can achieve high compression ratios with user specified distortion rates become necessary. Boundaries and edges in the tissue structures are vital for detection of lesions and tumors, which in turn requires the preservation of edges in the image. Unlike existing lossy transform-based compression techniques such as FFT and DCT, edge preservation is addressed in this new compression scheme. The proposed Edge Preserving Image Compressor (EPIC) combines lossless compression of edges with neural network compression techniques based on Dynamic Associative Neural Networks (DANN), to provide high compression ratios with user specified distortion rates in an adaptive compression system well-suited to parallel implementations. Improvements to DANN-based training through the use of a variance classifier for controlling a bank of neural networks speed convergence and allow the use of higher compression ratios for “simple” patterns. The adaptation and generalization capabilities inherent in EPIC also facilitate progressive transmission of images through varying the number of quantization levels used to represent compressed patterns. EPIC was able to achieve average compression ratios of 7.26:1 with an averaged Average Mean Squared Error (AMSE) of 0.0147.

Acknowledgements

I would like to acknowledge the suggestions and guidance given by my thesis advisor, Dr. Kabuka, as well as the members of my thesis committee, Dr. Banerjee and Dr. Jeon, for their input on how to improve the thesis. I would also like to thank Andres Rios for the valuable discussions regarding DANN and backpropagation training.

In addition, I am grateful for the help that Kevin Chase has rendered in proof-reading and correcting grammatical errors; I am responsible for any that remain. I am also thankful to my family and friends for their moral support during this time.

Table of Contents

Table of Contents iv

Chapter 1. Introduction 1

Chapter 2. Survey of Previous Work 6

2.1. Conventional Data Compression Techniques 6

2.1.1. Lossless Techniques 6

2.1.2. Lossy Techniques 8

2.1.3. Lossy With Lossless Residual Encoding Techniques 11

2.2. Data Compression Using Neural Networks 11

2.2.1. Neural Network Implementation of Vector Quantization 12

2.2.2. Cottrell/Munro/Zipser Technique 12

Chapter 3. Edge Preserving Image Compressor (EPIC) Architecture 15

Chapter 4. Extension of DANN Architecture & Training Algorithms for EPIC 20

4.1. Variance Classifier Selector for Controlling a Bank of DANN Networks 21

4.2. Thresholded Training Set Generation Procedure 21

4.3. Modified DANN Training Procedure 22

Chapter 5. EPIC Compression Algorithm 27

Chapter 6. Quantitative Measures of EPIC Performance 29

6.1. Compression Ratio of the EPIC System 29

6.2. Image Quality Error Measures 30

Chapter 7. Design of Experiments and EPIC Compression Results 32

7.1. Experimental Procedure 32

7.2. Compression Results 35

7.3. Discussion of Results 39

Chapter 8. Summary 42

8.1. Advantages of the EPIC Compression Scheme 42

8.2. Future Enhancements 43

Appendix 44

A.1. Dynamic Autoassociative Neural Networks — Architecture & Training 44

A.2. Backpropagation Formulae 47

References 49

Chapter 1. Introduction

Image compression is a field that is growing exponentially, due to its widespread applications in digital imaging and transmission. With the evolution of digital storage of visual information from grayscale still images to full-color real-time video images, ever increasing bandwidths will be needed if very high compression ratio real-time image compression techniques were not available. Image compression serves to alleviate this problem by reducing the required bandwidths and allowing new applications, such as videoconferencing, to coexist with existing data and information transmission on computer networks.

With the development of filmless radiology, where medical images are digitized and kept in electronic archives for instant access and display on graphical computer displays, the need for effective data compression techniques becomes even more important. This is due to the volume of image data that is generated every day, as well as the need to archive images over extended periods of time. The storage requirements for a medium sized hospital is expected to reach a few Gigabytes per year.

Nonetheless, existing image compression techniques are able to achieve compression ratios from 2:1 to 20:1 for most applications. The higher compression ratios have been achieved using “lossy” techniques that remove visual information that is not perceived by the human eye. Higher compression ratios can only be achieved at the expense of higher distortion that degrades the quality of the image. This is due to the general architecture of existing compressors, which are designed to be general enough to accommodate a variety of input characteristics. This trade-off means that those compressors are unable to take advantage of specific characteristics of the inputs to achieve better compression ratios while maintaining the same or lower levels of distortion.

To meet the challenges of achieving even high compression ratios, adaptive compression schemes have been developed. They are capable of adapting to the characteristics of the inputs as they vary. In this way, the image quality is preserved even though compression ratios are much higher. It is the focus of this thesis to determine the necessary architecture for implementing a successful compression system that meets the objectives of high compression ratios and low distortion rates by using an adaptive technique to match the compression system to the characteristics of the data. In addition, the suitability of such systems for the compression of medical images will be addressed as part of the research. The emphasis in medical applications is to preserve important image features that contribute towards accurate diagnoses. Therefore, edge information and image textures are important components that must be preserved by the image compression system.

The requirements for different data compression systems are tied inherently to the specific needs of the given application. For some applications, such as the compression of computer programs, no distortion or changes in the data can be tolerated, and this requires the use of lossless compression techniques. Ideally, all data compression schemes should be lossless, since this would circumvent problems related to determining what constitutes non-redundant information. However, Shannon’s entropy relation indicates that the achievable compression ratio for general data is about 2:1 using lossless techniques. For data sources such as still images, audio, and video, these ratios are insufficient for coping with the bandwidth and storage requirements of thousands of such images. A simple uncompressed 256 x 256 sized image with 8 bits per pixel requires 64Kb of storage, whereas in applications such as High Definition Television and medical image archiving and transmission, thousands of images need to be processed, stored or transmitted. High compression ratios are therefore necessary for meeting those requirements.

From biological studies, it has been determined that our visual and auditory senses perform some “masking” of the received inputs, rendering a portion of the received input redundant and unperceivable. This enables us to devise lossy data compression techniques which remove this redundant information, thus achieving much higher compression ratios. The rate distortion theory details the tradeoff between achievable compression ratios and the resultant distortion incurred on the data source [1]. This is the rationale behind modern image compression techniques such as the JPEG still image compression scheme, which provides compression ratios of 5:1 to 20:1 with reasonable distortions. Nonetheless, most of these algorithms do not provide uses with much control over the specification of which components of the images are non-redundant, since the associated cost-functions are usually global in nature and affect the entire image or dataset. This presents a problem for applications such as the compression of magnetic resonance images, since not all image data is equally important for image interpretation and analysis.

The characteristics of an appropriate image compression scheme can be defined as follows:

(i) the ability to specify which components of the image are vital for visual integrity and contain the most information, and therefore, need to be preserved with very little or no loss.

(ii) the ability to achieve as high compression ratios as possible for the other portions of the image, leading to significant savings in transmission and storage requirements.

(iii) the need to control the distortion incurred by the high compression approach in (ii) to within user specified levels.

(iv) the ability to adapt to changes in the input data stream.

(v) the ability to perform the compression and decompression as fast as possible, for use in real-time applications.

For JPEG and other transform-based compression techniques, the cost-function is the energy or frequency content of the image, which is concentrated in the lower values. Compression is achieved through the elimination of the higher energy or frequency values, which are less concentrated and therefore deemed less important. However, the high energy or frequency components of the image correspond to the edge information — the use of transform-based coding techniques effectively results in the blurring of edges and other high-frequency contents of the image. Furthermore, at high compression ratios (which is controlled by the Q-Factor in JPEG), the distortion rate increases significantly as well, resulting in “patchy” or “blocky” images. The JPEG algorithm has been devised as a compromise for compressing a wide range of image types — as such, it does not adapt to the characteristics of the input source in order to improve the tradeoff between compression ratios and rate distortion. Real-time implementations of the Discrete Cosine Transform (DCT) used in JPEG are also costly, since DCT is computationally intensive.

Medical image compression, such as the compression of magnetic resonance images, requires the detection and recognition of boundaries in the tissue structures, for detection of lesions and tumors. This requires the preservation of edges in the image, which defines the various boundaries between the various anatomical structures and tissues in the image. In addition, the permissible distortion rate is zero to very low values for medical image compression due to its nature, resulting in a conflict between requirements (ii) and (iii), since distortion rate and achievable compression ratios are directly related (a high compression ratio implies a high distortion rate). Adaptation (requirement (iv)) is an offshoot of requirements (ii) and (iii), since a high compression ratio implies that the system is very efficient in its ability to convert uncompressed input into compressed output. Since actual data sources exhibit changes in their characteristics over time, adaptive compression schemes are necessary for maintaining the high compression ratios and specified distortion rates. Real-time compression and decompression systems will be required in the future, when interactive image reconstruction and 3D modeling is required for assistance in patient studies, education and research.

Several of the requirements (ii)-(v) are well-suited for neural network based solutions. Neural networks are adaptive in nature, due to their use of a training phase for learning the relationships between the input data and required outputs, and their generalization capabilities provide a means of coping with noisy or incomplete data; while new neural networks presented in [2,3] provide a solution to the problem of achieving high compression ratios at very low distortion rates.

In order to achieve the goals of developing a suitable image compression architecture for magnetic resonance images, the following areas will be addressed in this thesis:

i) The preservation of edge information in the recovered image with no distortion.

ii) The use of an image classifier to separate image blocks into different classes and to compress the different classes of images using DANN-based networks that are optimally trained for each class.

iii) The improvement in the training set generation criteria, to eliminate redundant patterns and speed training.

iv) The extension of the DANN-based neural network training approach to improve the speed of convergence during training.

v) The combination of the different areas into a compression system.

The following chapters outline the details for the Edge Preserving Image Compressor (EPIC), beginning with a survey of existing compression techniques, the high level specification of the compressor architecture, the methods used to extend DANN-based compression for EPIC, the EPIC compression algorithm, and the results of using EPIC for compressing magnetic resonance images.

Chapter 2. Survey of Previous Work

Image data compression and decompression can be classified into two basic categories, lossless compression and lossy compression. Data compression is alternatively termed source coding for the case of lossless compression, since it entails finding the most compact representation of the data source. In general, lossless techniques are first generation techniques which utilize algorithmic approaches to compressing data. Lossy techniques usually adapt to the characteristics of the human visual system for determining what information is visually important.

2.1. Conventional Data Compression Techniques

2.1.1. Lossless Techniques

Lossless coding is based on the work in Information Theory by Shannon. The major categories of lossless codes include: Huffman coding, Arithmetic coding, Run Length Coding, and Markov source coding.

1) Huffman coding: To encode source data in the most compact form possible, variable length codes based on the probabilities of each source symbol in a memoryless source are used. Huffman coding is an optimal way of generating such variable length codes [4]. However, Huffman coding assumes a static source model, in which the probabilities of each image component remain constant. This limitation may actually result in source expansion if the characteristics of the image to be encoded are vastly different from that of the source model [5].

2) Arithmetic coding: Arithmetic coding, and its derivative technique, Q-coding, is used to overcome some of the limitations of Huffman codes. It is a non-block code, in that a single codeword is used to represent an entire sequence of input symbols, in contrast to

Huffman coding where a source symbol block corresponds to a codeword block. Instead, it uses the real numbers to represent a sequence of symbols by recursively subdividing the interval between 0 and 1 to specify each successive symbol. The limitation of this technique is the precision required in performing the calculations and arriving at the code word which will represent the entire sequence correctly.

3) Run Length Coding: If we view the image as a sequence of bits, we can often detect long runs of zeros framed by ones, due to the presence of large uniform regions in most images. Run Length Coding tries to exploit such inherent uniformity by using numbers to count the runs of zeros. The disadvantage of Run Length Coding is that it only deals with 1D correlation within the image, and ignores spatial (2D) correlation.

4) Markov source coding: In order to deal with the dependencies often present in an image where a large number of pixels are highly correlated (termed Markov sources), coding schemes such as Lempel-Ziv-Welch (LZW) and its variants have been developed which build a dictionary of the input source sequences. Compression occurs by the fact that successive occurrence of the same sequence will be represented by pointers to existing dictionary entries.

These first generation techniques achieve an average compression ratio of 2:1 to 3:1. The performance of such methods is insufficient for dealing with the storage requirements of future imaging systems, partly due to the fact that they are 1-D algorithms that treat the data as being a bit stream and are unable to take advantage of spatial correlation of image data in 2D or 3D. Schemes that take advantage of such correlation include bit-plane processing, predictive coding and Huffman encoding of differential images [5,6]. However, the lossless attribute of such schemes is viewed as being advantageous in the event that the raw data can be reconstructed for use without any change in the data.

2.1.2. Lossy Techniques

Lossy techniques all involve the elimination of certain image components in order to obtain a more efficiently coded representation of the image [7]. The major division of the different types of lossy compression techniques are Pulse Code Modulation (PCM) derived schemes, Transform Coding schemes, Block Truncation schemes, Vector Quantization schemes, and Sub-band Coding schemes. In addition, hierarchical coding schemes which utilize components from the various techniques have been proposed to improve the compression ratios or reduce the processing complexity of a given technique. A good survey of several of the above schemes can be found in [8,9].

1) Pulse Code Modulation: The PCM derived schemes work by using a predictor function in order to determine the value of the current pixel based upon the values of previously encoded pixels. The simplest predictor uses a 2 level quantizer and is termed Delta Modulation. It has the disadvantage of not being able to represent quick changes in the signal due to the limited step size of the quantizer. In order to handle larger variations in the signal, Differential PCM (DPCM) can be used, with the predictor having more than two levels. This is one of the more successful implementations of PCM schemes, in that it can be modified to account for 2D spatial correlation in order to increase the data compression ratio. Adaptive DPCM schemes have achieved compression ratios of 3.5:1 [5].

2) Transform Coding: Among the various lossy schemes, the most popular have been those based on Transform Coding. These includes the Karhunen-Loeve transform (KLT), Discrete Fourier transforms, Walsh–Hadamard transforms and Discrete Cosine Transform (DCT) (utilized in the JPEG standard), which all attempt to reduce the image correlation in order to represent it in as few basis functions as possible.

2a) KLT gives the minimum distortion for any given image, but it is never used in practice due to its computational complexity. For every image, an autocovariance matrix has to be evaluated together with its eigenvalues and eigenvectors. However, the KLT represents the upper bound for comparing compression efficiency of different techniques.

2b) Fast Transforms: The following transforms are linear and can be decomposed into fast variants that have $O(N \log_2 N)$ complexity instead of the normal $O(N^2)$ associated with the original transforms. Fourier Quantization [10] is a transform technique which has been investigated for digitizing hand images. In contrast, DCT is a fast but suboptimal transform which operates on a small section of the image at a time. This invariably results in “blocking” effects where subimages show sharp transition artifacts or block-like appearance at the edges of each subimage. In addition, transform techniques involve a lot of computationally intensive steps during compression and decompression, making software implementations cumbersome for real-time applications. However, dedicated processors have started to appear for performing the transform manipulations quickly. Variations of the DCT technique that are documented in [11–13] attempt to improve its performance and reduce the blocking artifacts. One approach is to perform the DCT on the whole image as a single frame [13]. This alleviates the blocking effect at the expense of compressor complexity and memory requirements. Acceptable compression ratios of 4:1 for MR images to 20:1 for CR chest images were recorded. Adaptive DCT (ADCT) has been investigated, which uses a variable encoding scheme for the transformed image, allocating more bits to coefficients with higher values [14]. This improves the compression ratio by 25-30%.

3) Vector Quantization: Another approach utilizes Vector Quantization techniques [5] for the lossy compression of MRI images [15]. Techniques based on Vector Quantization have asymmetrical processing requirements. Decoding is greatly simplified due to the use

of a Vector Codebook which is used to reconstruct the image. However, the computational overhead involved in selecting optimal vectors for encoding the images is quite high, as well as the choice of the codebook itself. Variations of the technique can be used to provide for image error correction and distortion control capabilities using variable-sized image blocks [16] as well.

4) Subband Coding: Subband coding with DPCM quantization and post quantization compression [17] has been utilized for image compression of non-medical images. It utilizes the characteristics of the human visual perception system to filter out portions of the image spectrum to which the eye is insensitive. The technique has achieved very good compression ratios, with the JPEG standard test image achieving up to 0.23 bits/pixel.

5) QuadTree Based Techniques [18,19]: Other compression techniques exploit the structure of the images in order to improve the compression ratio. By sectioning the image into regions which have similar characteristics, the complexity of representing each region effectively is reduced. Examples of such techniques are QuadTree representations and Region of Interest specification in the image. QuadTree Based techniques divide the picture recursively into quadrants and achieve compression by encoding a uniform quadrant by an average pixel value. Details are preserved by subdividing quadrants until the pixel values can be encoded using the average value described previously. This is an approach which does not require complex transformation techniques.

6) Block Truncation Based Techniques [5]: This technique, termed Block Truncation Coding (BTC), divides the image into small blocks, and then represents the block by the average value and standard variation of the pixel values obtained through a moment preserving function. In addition, a bitmap obtained by thresholding using the average value is then generated for the block. This effectively reduces pixel values to single bits,

with the additional overhead of storing the mean and variance of each block. More recently, there have been efforts to combine the advantages of BTC, VQ and DCT in a hybrid approach [20]. The achieved compression ratio is about 10:1.

2.1.3. Lossy With Lossless Residual Encoding Techniques

1) DPCM with Residual Image Coding [5]: Since the ability to reconstruct the original image may be extremely important if advances in medical image processing enabled better diagnosis using the raw data, the ability to perform lossy encoding with residual encoding of the errors will be extremely important. This technique basically compares the raw image with the compressed image to obtain the residual (error) image. The residual image is stored and can be used to reconstruct the original image with reduced storage requirements. DPCM with residual image coding is such an example.

2) Progressive Transmission (Hierarchical Interpolation) [5]: Another strategy to reduce transmission and retrieval time for initial clinical use is the hierarchical approach where the original image and a series of compressed images with increasing compression ratios are stored in an image hierarchy. The image with the lowest acceptable compression ratio is transmitted first for reference, and the original image is transmitted later for primary diagnosis. This approach trades off transmission time for storage requirements.

2.2. Data Compression Using Neural Networks

Neural networks are highly parallel computational systems with simple computational elements called neurons arranged in hierarchical layers. Each neuron has a set of inputs with individual weights associated with each input. The output of each neuron is a non-linear response of the weighted sum of the inputs. Neural networks undergo “training” in order to condition the outputs to a desired response when presented with an input. Training proceeds by measuring the error present at the output between the actual and desired responses. This error is then used to adjust the weights associated with each input, and hence the network adapts until the outputs converge to the desired response for a given input. After a sufficient amount of training, the neural networks can then be used for the compression of images not used for training. The adaptability of the output response to accommodate unknown inputs is the key to the success of neural network based techniques. Compression is achieved by the means of multi-layered neural networks that have small hidden layers (in comparison to the output layer) [21].

2.2.1. Neural Network Implementation of Vector Quantization

A recent work using neural network approaches to image compression is based on Vector Quantization. Since the selection of the appropriate vectors for representing a given image is basically a classification problem, neural networks which have been trained to do so optimally will greatly increase the compression speed. Nevertheless, reported results still indicate a rather high distortion rate, with a signal to noise ratio (SNR) of 26.9 for a 20:1 compression ratio [22]. Further improvements in the error rates are necessary before this technique can be applied to solve medical image compression problems.

2.2.2. Cottrell/Munro/Zipser Technique

Most of the neural network based techniques are derived from the work by Cottrell, Munro and Zipser [23]. They developed a multilayered perceptron neural network with back propagation as the error learning function.

1) Back-propagation with Multi-level Neural Nets: From [21], Multi-Level Neural Nets are trained using back-propagation with the block sampled original image at the input and output in order to obtain a reduced vector representation of each block in the hidden layer. The internal (hidden) representation of the block is the compressed form of the image to be transmitted since there are less hidden layer units than at the input or output.

By applying the internal vector to the decompressing network, the original image can be recalled. This approach uses a form of self-organization for obtaining the reduced dimension feature space. This is similar to Principal Component Analysis in its mode of operation. [24,25] are descriptions of systems which utilized this technique for data compression in non-medical environments. Neural networks are trained using a set of sample images, and then the weights obtained through the training phase are used to compress actual images.

2) Non-quadratic Error Functions for Back-propagation: It has been shown that back-propagation training using quadratic error functions are shown to be equivalent to performing KL transforms on the image [26]. Improvements in image restoration using non-quadratic error functions and decreasing convergence time by constraining the neural network weights were proposed as solutions to the inherent shortcoming of classic back-propagation based training [26]. The traditional measure of success in training neural networks is based on the mean square error (MSE) observed at the output between the training images and the output of the neural network. However, it is found that MSE is a very unreliable measure for determining the correctness of the weights used for compression, since the output error (from observing the decompressed images) can be significant when the global error (as measured using MSE during the training phase) is low [27]. Data compression ratios of 8:1 were achieved. The solution to this dilemma was to use a new training scheme called the epsilon descent technique.

3) Adaptive Hidden Layers: Other attempts to improve the compression results include using neural networks that have a variable number of hidden units that adapt during the training process in order to escape local minima in the error function. Proper selection and training of the neural networks is crucial for its success. This architecture can adapt

to different input data characteristics through its adaptive hidden layer which provides the necessary learning support to achieve convergence in the training process.

4) New neural network based adaptive compression methods which overcome the traditional drawbacks associated with back-propagation based neural networks, such as the static nature of network architectures, unbalanced errors of individual training patterns, and being trapped in local minima during training, have been developed [2,3]. This new neural network architecture, termed Dynamic Autoassociative Neural Networks (DANN) has been shown to provide excellent control over the error rates while maintaining user-selectable compression ratios.

Chapter 3. Edge Preserving Image Compressor (EPIC) Architecture

The proposed Edge Preserving Image Compressor (EPIC) architecture is a new hybrid image compression technique combining conventional techniques with DANN-based neural networks. EPIC utilizes techniques that are similar to the approach of the edge/non-edge compression algorithm proposed in [28], which in turn is derived from the synthetic-highs compression technique. However, instead of the adaptive DCT compression used for the non-edge image as is done in [28], neural network based compression is utilized, and the edge-subtraction step is eliminated entirely. It is designed to be modular in nature, and improvements in each module can be incorporated easily into the architecture, resulting in better accuracy and performance of the system. One of the major criteria of this architecture is the preservation of edge information of the original image. Since edge detection is a vital step in any image processing and manipulation task, the inclusion of this step into the image compression architecture results in two distinct advantages: the elimination of the need for edge detection in subsequent downstream processing of the images, and the guarantee that edge information is not lost during compression, unlike many transform-based methods. The edge information extraction is useful for other reasons as well. Correlation of anatomical features between images obtained using different modalities are vital for various diagnostic functions involving tissue identification, classification, and metabolic modeling. For instance, CT and MRI image features are used to localize anatomical features obtained from PET and other modalities with less distinct image features and boundaries. Examples of such applications can be found in [29–31].

Since edge information has been preserved, we can compress the rest of the image at high compression ratios with a user-controlled distortion rate without significantly affecting the overall picture quality of the reconstructed image. Nonetheless, for certain applications such as medical imaging, very minimal or no loss compression is required where image integrity is of utmost importance. The somewhat contradictory goals of high compression ratios and low error rates therefore dictates the use of an adaptive compression technique which could satisfy those two constraints. Neural network based techniques, which have shown significant advantages over other techniques in their adaptation and generalization capabilities, have been selected for this purpose. The neural networks are trained to compress image features present in specific types of images. By targeting a specific data compression domain for the neural network compressor, the architecture aims to achieve both objectives through the effective use of adaptation and generalization capabilities inherent in neural networks.

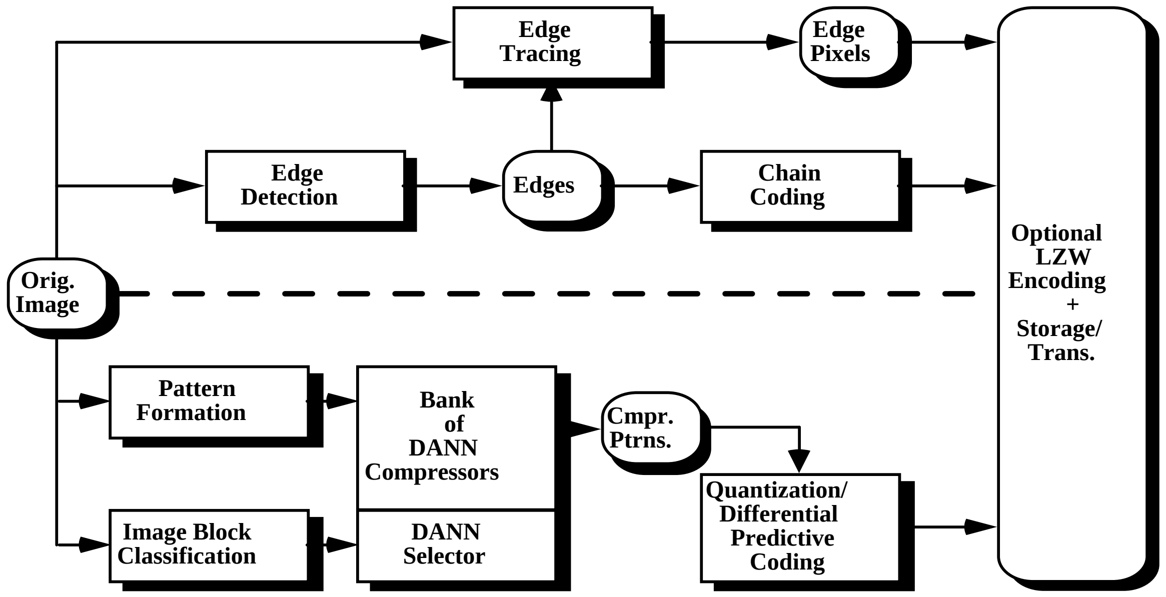
The second goal of the compression architecture is to create a system that is easily converted to parallel implementations. Neural network systems are well suited to such implementations due to their uniform structure and the distributed nature of the neural computation, output, and weight adaptation processes. Hardware implementation of neural network topologies will lead to the development of compression systems that can perform their tasks in real-time. The edge detection step can be executed in parallel with the neural network based image compression process since the edges are coded independently using chain-coding. This enables us to select the most appropriate edge detection technique for the given images, while improvements in edge detection algorithms and changes in image integrity criteria can be “plugged into” the compression system without affecting the rest of the compression process.

Another means of parallelizing the image compression process is through the use of a bank of neural networks for the compression process. The image is subdivided into uniform sized blocks that are fed to a bank of neural networks. Each network has been previously trained for a specific class of image data based on the variance value of the image block. The appropriate network is selected for the given image block, and compression of each block proceeds independently of the other blocks. It is therefore possible to pipeline and parallelize this process for the compression of blocks belonging to different classes, achieving significant improvements in processing speed. The advantage of using a bank of neural networks tailored to specific image classes is the capability for parallelism during the training process. The entire training set is divided over the bank of networks, and the number of patterns that each network is required to learn is therefore reduced, or alternatively, the system is capable of learning more images, improving its adaptation capabilities.

Among the various edge detectors that have been developed, the Second Derivative Exponential Filter (SDEF) [32] and the Canny filter [33] have proven to be very accurate in terms of edge localization. However, in empirical use, the SDEF filter tends to enhance noise effects, and a noise removal stage using a speckle filter is necessary for eliminating spurious and noisy edges. In contrast, the Canny filter, with its adjustable window size and edge thresholds, produces very clean edges without the need for noise removal. EPIC provides the option of selecting either Canny or SDEF edge detection schemes during the compression phase. A classifier network examines each block of the image and determines which of a bank of neural nets previously trained is best suited for the compression task using a simple variance classifier. The ability of the neural network to generalize and be trained to adapt to specific types of patterns in the input image is expected to yield better compression ratios and image fidelity as compared to compression using conventional DCT techniques, and is inherently more parallelizable in hardware implementations due to its uniform structure and simple processing requirements. The neural network architecture utilized in EPIC is a multi-layered Perceptron network with two hidden layers, which has been used successfully for image data compression in [2,3]. The input vector is compressed into a neural network pattern representation using a compression ratio dictated by the ratio between the number of input neurons (equal to the number of pixels in the image block) and the number of neurons in the second hidden layer. The compressed pattern outputs from the neural network compressor is then coded using either a variable bit-rate linear predictive coder, or quantized using a fixed bit-rate with marginally higher distortion. Optional Lempel-Ziv-Welch (LZW) encoding is applied to the outputs of both the edge coding and neural network compressors to remove any residual redundancies. These patterns are then stored or transmitted for use in reconstructing the images at the decompressor.

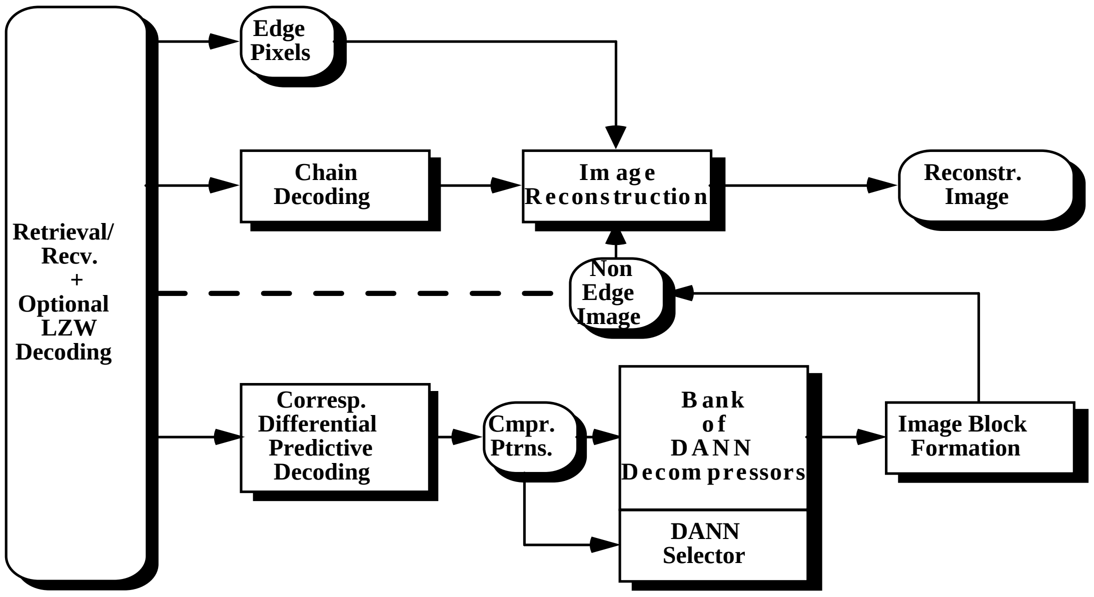
The decompressor performs the corresponding optional LZW decoding on the received edges and compressed patterns, as well as corresponding linear predictive decoding on the compressed patterns (if required), and passes it through a bank of neural networks which converts the compressed patterns into recovered output blocks that are used to form the non-edge image. This non-edge image is then overlaid with the decoded edge pixels to reconstruct the reconstructed image.

The compression system can be shown in the following diagrams (Figures 3.1 and 3.2):



Edge Preserving Neural Network Based Image Compressor

Figure 3.1: Compression Overview



Edge Preserving Neural Network Based Image Decompressor

Figure 3.2: Decompression Overview

Chapter 4. Extension of DANN Architecture & Training Algorithms for EPIC

Dynamic Associative Neural Networks (DANN) have been used effectively for image compression [2,3]. Consequently EPIC utilizes DANN-based neural networks for compressing the non-edge image. A brief summary of DANN characteristics is provided in the Appendix. However, DANN training suffers from the problem of slow convergence, since the descending epsilon training technique [27] is a very slow process. There are often specific input patterns that are considered very difficult to learn. The single network used for DANN compression to compress all possible input patterns therefore has to account for those difficult patterns in its learning process as well. This further increases the required network capacity and complicates the learning process since the network often has to adapt to input patterns with very different characteristics. All these factors limit the ability to extend DANN-based compression techniques to provide very high data compression ratios while providing very low error rates for use with application where data fidelity is extremely important. EPIC improves upon DANN-based compression through the following features:

i) The use of a bank of DANN networks in place of a single network for processing the incoming data stream, under the control of a variance classifier.

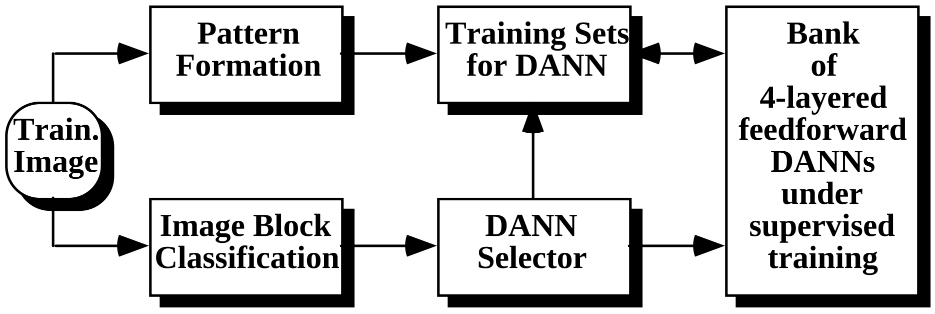
ii) Improvements in the training set generation procedure through the elimination of duplicate training vectors which are close together in Euclidean space, controlled by the use of a similarity threshold.

iii) Improvements in the training process, by specifying termination criteria for pathological training conditions, where the network error is increasing continually or is stuck for a given number of training epochs.

The changes to DANN architectures and training algorithms are detailed in the following sections.

4.1. Variance Classifier Selector for Controlling a Bank of DANN Networks

A bank of four-layered feedforward perceptron networks (similar to those used in DANN compression) is used to process the incoming data stream. The data is classified using a simple variance classifier into 1 of P classes, where P is determined heuristically from variance characteristics of the data, and compressed by the corresponding network. In this way, the cost of compressing complex patterns can be isolated from the cost of compressing the other patterns. The network used to compress a pattern with low variance will require a small first hidden layer and provide very high compression ratios, in contrast to other networks which have larger first hidden layers, and possibly lower compression ratios. The outline of the training process is given in Figure 4.1.


Pattern Formation
Training Sets for DANN
Bank of 4-layered feedforward DANNs under supervised training
Train. Image
Image Block Classification
DANN Selector

Neural Network Image Compressor Training Outline

Figure 4.1: Compression System Training Block Diagram

4.2. Thresholded Training Set Generation Procedure

The generation of the training set for training the bank of P neural networks is to take a set of training images, and partition each image into N x N sized blocks, overlapping by N/2 pixels, for use in generating the training set. The variance of the block is calculated for use in the classification step. The blocks are classified into one of P classes, based on predefined variance thresholds determined empirically through statistical analysis of the characteristics of typical images, and converted into a training vector. The candidate vector belonging to the p-th class is then compared with all existing training vectors in the training set for that class to determine if it exceeds a “similarity threshold” defined to eliminate redundant training vectors lying close together in Euclidean space. The user controllable “similarity threshold” serves to reduce the size of the training set and avoids overtraining through the use of highly similar training vectors, which reduces the generalization properties of the networks. This is illustrated in Figure 4.2.

```
For given set of training images
    Partition image into N x N blocks, which are overlapping by N/2
    For each block (parallelizable)
        classify the block into p    P classes using predefined variance thresholds
        compare block with all existing patterns in the p-th training set
        if the block exceeds the similarity threshold
            add block to the p-th training set
        endif
    endfor
endfor
```

Figure 4.2: Training Set Generation Pseudocode

4.3. Modified DANN Training Procedure

The neural network training algorithm used in EPIC is based on the DANN training procedure outlined in [2]. Each neural network in the bank of P networks is subjected to DANN training, with some modifications designed to detect pathological conditions and reduce time spent in unproductive training. Among the improvements are convergence detection primitives, which measure the slope of the error curve to determine if the training process is stuck (wobbling) or if the error function is increasing as training proceeds (increasing error). Thresholds are set for the number of iterations the network can exist in these conditions before that particular phase of training is aborted and control returned to the calling routine.

DANN training operates on two levels: the higher level is the generative process by which the network grows and adjusts the number of hidden nodes needed to learn the required mapping between the inputs and the outputs. This controls the dynamic aspect of the network. The lower level is based on backpropagation training, in which the network with a given hidden layer size undergoes modification of its weights to reduce the value of the error function and converge towards predefined error thresholds. Training is controlled by two parameters, Eta (learning rate), and Alpha (momentum). The error thresholds are user-selectable and hence provide users with a known bounded error characteristic for the compression system. The backpropagation training is further refined through the use of the descending epsilon training approach [27], which forces the error curves for each pattern to within a set interval as the training progresses, resulting in a more even distribution of errors throughout the entire training set. In addition, Eta and Alpha are scaled to successively smaller values at each interval of descending epsilon training to provide better convergence characteristics. Validation of DANN training is done using a different set of patterns than those used for descending epsilon training, to provide an objective measurement of the training performance.

The modified descending epsilon training process is given in Figure 4.3. For each training pattern, it is propagated through the network and the outputs compared to the desired outputs for computing the error values for each output node. If the absolute error is less than the Epsilon parameter, it is set to zero, otherwise the error is kept. If all output errors are zero, that means the network has been trained correctly for that pattern and the number of correct patterns is incremented. Otherwise, the count is reset to zero and the adjusted errors backpropagated through the network for weight adjustment. If all the patterns in the training set passed the error adjustment test (i.e.: number of correct patterns is equal to the number of training patterns), then the epsilon parameter is decremented and the process repeated until the final epsilon value is reached or the network fails to converge within a specified number of training epochs. If the error function is wobbling or increasing, the epsilon training phase is aborted and DANN training resumes control.

```
Set Epsilon to Start Epsilon
While (not done)
    Get next pattern from training set
    Propagate pattern through network
    Adjust output errors using Epsilon parameter:
        For each neuron in output layer
            If (| Output – Expected Output | < Epsilon)
                Error = 0
            endif
        endfor
    If all output neuron errors = 0
        Increment Num. Correct Patterns
        If (Num. Correct Patterns == Num. Patterns in training set)
            If (Epsilon > Stop Epsilon)
                decrement Epsilon by EpsilonStep
                Scale Eta, Alpha by respective ratios
            Else
                done = 1
            endif
        endif
    Else
        Backpropagate errors
        Adjust network weights
        Reset Num. Correct Patterns
    endif
    Increment Pattern Counter
    If (not done) and (Pattern Counter == Num. Patterns in Training Set)
        Increment Num. of Epochs
        Validate Network, obtain avg. pattern error for network
        Calculate slope of Error function
        If (error function is wobbling)
            Increment Stuck count
            If (Stuck count >= Max. Stuck count)
                done = 1
            endif
        Else
            Reset Stuck count
        endif
        If (error function is increasing)
            Increment Error Increasing count
            If (Error Increasing count >= Max. Error Increasing count)
                done = 1
            endif
        Else
            Reset Error Increasing count
        endif
        if (Max. Epochs reached)
            done = 1
        endif
    endif
endwhile
```

Figure 4.3: Modified Descending Epsilon Training Pseudocode

Modified DANN training is given in Figure 4.4. This higher level training procedure begins by first defining an initial first hidden layer size and the maximum first hidden layer size which the network can reach before training is declared unsuccessful. DANN training is further divided into a Growing Phase, a Shrinking Phase, and a Final (cleanup) Phase. During the Growing Phase, the network is subjected to descending epsilon training, and if the network fails to converge to the specified target error, the first hidden layer size is incremented by *m* neurons and descending epsilon training resumed. Pathological conditions such as a stuck error function or an increasing error function are detected and training aborted (declared unsuccessful) in such cases. This helps the network designer by indicating that either the given first hidden layer size is insufficient for learning the given training patterns, and therefore the initial first hidden layer size needs to be increased, or that other parameters such as the learning rate and momentum need to be adjusted.

If the Growing Phase completes successfully, the network is then subjected to a Shrinking Phase to remove any redundant hidden layer nodes, since the network becomes overconstrained after undergoing the growth process. The Shrinking Phase is closely monitored by storing intermediate network states to ensure that the target error is still met if a marginal first hidden layer node is removed. If the target error is not met, the last network state which satisfies the error constraints is recovered from temporary storage and used in the Final Phase.

The Final cleanup is done to enable the network to settle into its new topology and possibly improve its error performance beyond the targeted values. Backpropagation training (without the use of descending epsilon training) is performed once on the network and this results in the successfully trained network.

The EPIC training process for each of the P networks can be done in parallel, decreasing the time required for learning a given set of images compared to the original DANN training process.

```
While (Network not converged) and (number of training iterations not exceeded)
Growing Phase:
    While (Num. first Hidden Layer neurons (NHid1) < Max NHid1)
            and (Network not converged) and (not incrdone)
        Perform Descending Epsilon training
        Validate Network using Validation Set, obtain avg. pattern error for network
        Calculate slope of Error function
        If (error function is wobbling)
            Increment Stuck count
            If (Stuck count >= Max. Stuck count)
                incrdone = 1
            endif
        Else
            Reset Stuck count
        endif
        If (error function is increasing)
            Increment Error Increasing count
            If (Error Increasing count >= Max. Error Increasing count)
                incrdone = 1
            endif
        Else
            Reset Error Increasing count
        endif
        If (not incrdone) and (NHid1 < Max NHid1) and (Network not converged)
            Increment first hidden layer by m neurons
            Reset Eta, Alpha
        endif
    endwhile
Shrinking Phase:
    Store Network state
    While (Network convergence criteria met) and (NHid1 > 1)
        Decrement first hidden layer by one neuron
        Reset Eta, Alpha
        Perform Descending Epsilon training
        Validate network using Validation Set
            (check avg. net error, network convergence criteria)
        if Validated, Store new Network state
    endwhile
Final Phase (cleanup):
    Restore last valid Network state
    Perform Backpropagation training
    Store Network state
endwhile
```

Figure 4.4: Modified DANN Training Pseudocode

Chapter 5. EPIC Compression Algorithm

The EPIC compression technique is essentially an asymmetric compression scheme. The processing overhead is incurred at the compression phase, due to the edge detection and block classification steps. The decompression phase only requires the reconstruction of the non-edge image and the overlaying of the edge pixels onto that image to result in the reconstructed image.

An incoming image of size n x n is passed through an edge detector for extracting the edge pixels for lossless chain coding while simultaneously being subdivided into N x N non-overlapping blocks for processing by the neural network based lossy compressor. The block is classified into 1 of P classes. This process of decomposing the image into individual blocks for use in classification can be performed in parallel. The classified blocks are then fed to the bank of neural networks for compression. Each network has been trained previously using image blocks that have the same variance characteristics and is thus well adapted for those blocks. The second hidden layer compressed patterns are then extracted, and either coded using a zeroth order Linear Predictive Coder at a variable bit rate, or alternatively, quantized at a fixed bit-rate with marginally higher distortion. Associated with each compressed pattern is the network class for use in decompression. The chain coded edge image, as well as the compressed patterns, are optionally further compressed using the LZW-based UNIX compress command to remove any further redundancies in the data stream. This usually results in no more than 10% improvement in the compression ratio, however, and is obtained only from the chain codes. It is conceivable that this step could be eliminated with the use of a Huffman coder for the chain codes.

Decompression is relatively straightforward. The compressed non-edge pattern and edge image are first decompressed using the UNIX uncompress command as necessary, and the compressed patterns extracted and fed to the decompressors based on their network class for decompression into the respective image blocks. The block decompression step can be performed in parallel. The various blocks are used to form the non-edge image. In addition, the chain coded edge image is decoded and overlaid onto the non-edge image to give the reconstructed image.

The algorithms for compression and decompression are given in Figures 5.1 and 5.2.

```
For given set of images
    Perform Edge Detection on the image
    Store Edges and their Gray values using chain coding
    Partition image into N x N non-overlapping blocks
    For each block (parallelizable)
        classify the block into p    P classes using predefined variance thresholds
        compress the block using the pth neural network
        Quantize output of second hidden layer
    endfor
    Store compressed patterns using differential linear predictive coding,
        or fixed-bit rate quantization for higher compression ratio
    Optional edge compression stage using LZW encoding for storage/transmission
endfor
```

Figure 5.1: EPIC Image Compression Pseudocode

```
For given set of images
    LZW decoding (correspond to optional encoding) of edge compressed images
    Decode chain of Edges and their Gray values
    Perform differential linear predictive decoding of compressed patterns (as needed)
    For each block (parallelizable)
        decompress block using the p-th neural network specified in pattern’s class info.
        Form N x N non-edge image block
    endfor
    Overlay Edge pixels on non-edge image to obtain reconstructed image
endfor
```

Figure 5.2: EPIC Image Decompression Pseudocode

Chapter 6. Quantitative Measures of EPIC Performance

6.1. Compression Ratio of the EPIC System

Since the image is decomposed into edge and non-edge portions, we can define compression ratios for the non-edge image only (NE-CR), the edge information only (E-CR), as well as the combined compression ratio (CCR), defined as the ratio of the size of the original image to the combined sizes of the edge information and non-edge image.

The combined compression ratio for an image (CCR), given a bank of neural networks which can have different compression ratios for each network, is given by:

$$CCR_{image} = \frac{Size_{orig}}{Size_{compr}}$$

$$Size_{orig} = n^2 b_{pixel}$$

$$Size_{compr} = Size_{edge} + Size_{patterns}$$

$$Size_{patterns} = \sum_{i=1}^{P} N_{h2_i} b_{h2_i} k_i$$

where

n^2: number of pixels in the image (dimensions $\mathrm{n} \times \mathrm{n}$)

N: dimension of each image block

K: total number of blocks per image. $K = \frac{n^2}{N^2}$

b_{pixel}: size of pixel in bits

$Size_{edge}$: size of edge pixel data (chain - coded graylevel edges)

$Size_{patterns}$: size of compressed pattern data

N_{h2_i}: number of second layer hidden neurons in network i

b_{h2_i}: bits per compressed channel symbol for network i

$k_{i:}$: number of blocks processed by network i, where $\sum_{i=1}^{P} k_i = K$

If we let $N_{h2_i} = N_{h2}, b_{h2_i} = b_{h2}$, we get:

$$Size_{patterns} = N_{h2} b_{h2} \sum_{i=1}^{P} k_i = N_{h2} b_{h2} K$$

which is the case where all networks have the same compression ratios

However, since the weights interconnecting the second hidden layer and the output layer have to be stored or transmitted to the decompressor in order to recover the image, the compression ratio for the system will then have to incur the cost of storing or transmitting those weights once, for a series of m images compressed using those weights:

$$CR_{system} = \frac{m \times Size_{orig}}{m \times Size_{compr} + Size_{weights}}$$

$$Size_{weights} = \sum_{i=1}^{P} b_{weight} N_{out} N_{h2_i} = b_{weight} N_{out} \sum_{i=1}^{P} N_{h2_i}$$

where

m: number of images compressed using the current weights

b_{weight}: bits per weight value

N_{out}: number of neurons in the output layer. $N_{out} = N_{in} = N^2$

Therefore

$$CR_{system} = \frac{mn^2 b_{pixel}}{m\left(Size_{edge_j} + b_{h2} \sum_{i=1}^{P} N_{h2_i} k_i\right) + b_{weight} N_{out} \sum_{i=1}^{P} N_{h2_i}}$$

where

$Size_{edge_j}$: size of edge image for image j (of m)

Note that these values do not include the slight overhead incurred by the use of descriptive file headers.

6.2. Image Quality Error Measures

The error measures used in the discussion of the results are the Mean Squared Error (MSE) and the Peak Signal to Noise Ratio (PSNR), two common qualitative measures used for comparing accuracy of compression systems [1]. They are defined as:

$$MSE = \frac{\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\left(x_{ij} - x_{ij}^{'}\right)^2}{\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} x_{ij}^{\ 2}}$$

$$PSNR = 10\log_{10}\left(\frac{n^2\,(\max\ graylevel)^2}{\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\left(x_{ij} - x_{ij}^{'}\right)^2}\right) dB$$

where

x_{ij} : original pixel value at (i, j)

$x_{ij}^{'}$: reconstructed pixel value at (i, j)

max *graylevel* : maximum possible graylevel value for the image

Chapter 7. Design of Experiments and EPIC Compression Results

7.1. Experimental Procedure

A set of MR images, comprising of a series of 24 images, were used for the training and compression tests. The images were numbered from 00 to 23, categorized roughly into head images (mri00 to mri11), and chest images (mri12 to mri23). The Canny edge detector was used for all the compression experiments as it was found empirically to provide much better defined and cleaner edges than the SDEF filter.

The training and validation images were partitioned into 16 x 16 blocks, overlapping by 8 pixels horizontally and vertically, for creating the training and validation patterns. The similarity threshold was set equal to 50.0, defined as the squared Euclidean distance between the candidate vector and existing vectors in the training set. This helped eliminate a lot of similar vectors which constitute non-image areas in the MRI (since the actual image area is a circle inscribed within the square image frame). Each bank of DANN networks contained eight networks, which were selected via the variance classifier network selector using heuristics based on variance thresholds defined through previous statistical analysis of the MRI training set. The variance of a block was calculated with respect to the values of pixels within the block. Each block is classified based on three variance thresholds, the first defined for the entire block, the second defined based on quarter blocks, and the third defined based on sixteenth-blocks. The thresholds were set equal to 10.0, 90.0, and 300.0, and were utilized by the variance classifier to select one of the eight DANN networks for subsequent training and compression. The DANN networks were numbered 0 to 7 for the various variance classes, with Network 0 corresponding to Class 1, and Network 7 corresponding to Class 4. The heuristic given in Table 7.1 was used to perform the variance classification:

Variance	**Class**	**Ntwk**
Entire block variance < 10.0	1	0
Block variance > 10.0, all 1/4-block variances < 90.0	2	1
Block variance > 10.0, one 1/4-block variance > 90.0	2q	2
Block variance > 10.0, two 1/4-block variances > 90.0	2h	3
Three or four 1/4-block variances > 90.0, all 1/16-block variances < 300.0	3	4
Three or four 1/4-block variances > 90.0, one to four 1/16-block variances > 300.0	3q	5
Three or four 1/4-block variances > 90.0, five to eight 1/16-block variances > 300.0	3h	6
Three or four 1/4-block variances > 90.0, more than eight 1/16-block variances > 300.0	4	7

Table 7.1: Variance Classifier Neural Network Selector Heuristic

The compression ratios of each network, defined as the ratio of the number of neurons in the input/output layers to the number of neurons in the second hidden layer, were also defined based on the variance class. Higher compression ratios were used for variance classes 1 and 2, while lower compression ratios were used for variance classes 3 and 4 to handle the more complex image blocks.

For Experiment (Expt.) CTEST4, the training set was generated from the first ten images (mri00 to mri09), while ten images (mri10 to mri19) were used as the validation set during training. Remaining images (mri20 to mri23) were used as additional test images. The values of eta and alpha used during training were chosen to be small, in the range of 0.1 (eta) to 0.2 (alpha), after it was determined through training trials that larger values cause the network error to decrease rapidly at the beginning but do not converge sufficiently during later phases of the training process. The values of eta and alpha were also set to decay as descending epsilon progressed to enable the network to follow the backpropagation gradient descent more precisely. Epsilon_start was set to 0.50001 at the start of the descending epsilon training process, and decreased to 0.00001 (epsilon_end) in stepsizes of 0.05 (epsilon_step). The target Average Mean Squared Error (AMSE) was set to 0.01, with the maximum number of epochs for descending epsilon training set to 500, with validation performed every 20 epochs. The descending epsilon training process was aborted if the network error was stuck for 10 validation cycles, or 15 cycles with continually increasing errors. Validation for DANN training was performed every time a descending epsilon training cycle was completed. If the target AMSE was not reached, DANN training would change the first hidden layer size until the maximum size was reached, or if the network error was stuck for 10 validation cycles or 10 cycles with continually increasing errors.

It was observed that for network classes with lower variances, the training parameters had to be adjusted for slower convergence, since the low variability training vectors results in very low average network errors. The training parameters for Networks 0 to 2 were chosen as — eta: 0.1, alpha: 0.1, Epsilon_start: 0.0000101, Epsilon_end: 0.0000001, Epsilon_step: 0.000005. In addition, during the later stages of training for the remaining networks, the epsilon parameter had to be reduced to much lower starting values to enable the network to converge towards the actual errors. Otherwise, the images became too "smoothed" and details were lost. Nonetheless, not all networks were able to converge to the target error rates, especially for the higher variance classes. However, the convergence property for Network 0 (Class 1) was found to be superior, and that characteristic was exploited in increasing its compression ratio to 256:4 (64:1) to provide a higher overall compression ratio. The results for the modified Network 0 architecture are categorized under Expt. CTEST4_A. The trained EPIC neural-network topologies for Expt. CTEST4 and CTEST4_A are given in Tables 7.2 and 7.3. The number of neurons in the input and output layers are equal to 256 because images were subdivided into non-overlapping 16 x 16 blocks for compression.

CTEST4				
Ntwk	Input	First Hidden	Second Hidden	Output
0	256	100	8	256
1	256	70	8	256
2	256	250	8	256
3	256	231	8	256
4	256	213	16	256
5	256	211	16	256
6	256	218	16	256
7	256	186	16	256

Table 7.2: EPIC neural-network topology for Expt. CTEST4

CTEST4_A				
Ntwk	Input	First Hidden	Second Hidden	Output
0	256	90	4	256
1	256	70	8	256
2	256	250	8	256
3	256	231	8	256
4	256	213	16	256
5	256	211	16	256
6	256	218	16	256
7	256	186	16	256

Table 7.3: EPIC neural-network topology for Expt. CTEST4_A

7.2. Compression Results

For experiment CTEST4 and CTEST4_A, the compressed outputs were quantized to 8 bits resolution for storage. In the case of MR images, the chain-coded edge information constitutes a much larger proportion of the combined compressed image size (which includes both edge and non-edge data). This constrains the upper bound for achievable Combined Compression Ratios (CCR) to the compression ratio for the edges (E-CR). In Expt. CTEST4, the average compression ratio achieved for the non-edge image (NE-CR) was 20.9:1, while the average compression ratio for the edge pixels (E-CR) was 10.42:1. The average Combined Compression Ratio (CCR) was 6.95:1. The average AMSE of the combined image (edge and non-edge) was 0.0147, which is equivalent to a PSNR of 71.02 dB. It could be seen that edge preservation improved the fidelity of the reconstructed image over that of the non-edge image only.

In contrast, for Expt. CTEST4_A, The NE-CR for Expt. CTEST4_A was higher due to the use of higher compression ratios for Network 0, leading to an improvement in the CCR to 7.26:1, while the higher compression ratio did not increase the overall AMSE from that achieved in Expt. CTEST4. The results for the set of 24 MR images compressed using EPIC are presented in Table 7.4 and Table 7.5:

EPIC Compression Expt. CTEST4								
Image	#QBits	NE-PSNR	NE-AMSE	E+NE-PSNR	E+NE-AMSE	NE-CR	E-CR	CCR
mri8I00	8	70.46	0.0269	72.64	0.0217	21.02:1	9.86:1	6.71:1
mri8I01	8	70.46	0.0181	72.49	0.0148	20.86:1	9.39:1	6.47:1
mri8I02	8	71.49	0.0142	73.34	0.0118	21.40:1	10.73:1	7.15:1
mri8I03	8	70.33	0.0144	71.99	0.0122	21.02:1	10.49:1	7.00:1
mri8I04	8	71.27	0.0129	73.04	0.0108	21.13:1	12.08:1	7.69:1
mri8I05	8	70.89	0.0127	72.66	0.0107	21.02:1	11.42:1	7.40:1
mri8I06	8	70.12	0.0169	72.00	0.0140	21.63:1	11.43:1	7.48:1
mri8I07	8	68.49	0.0200	70.53	0.0163	21.24:1	11.20:1	7.33:1
mri8I08	8	66.47	0.0269	68.46	0.0221	21.86:1	10.51:1	7.10:1
mri8I09	8	68.22	0.0193	70.15	0.0159	21.63:1	10.85:1	7.23:1
mri8I10	8	68.64	0.0154	70.47	0.0128	21.35:1	11.16:1	7.33:1
mri8I11	8	69.94	0.0119	71.68	0.0100	21.24:1	10.84:1	7.18:1
mri8I12	8	68.53	0.0122	70.38	0.0101	20.39:1	10.32:1	6.85:1
mri8I13	8	68.50	0.0120	70.97	0.0094	19.90:1	10.38:1	6.82:1
mri8I14	8	67.86	0.0144	70.56	0.0110	20.19:1	9.54:1	6.48:1
mri8I15	8	68.57	0.0152	71.48	0.0114	20.09:1	9.44:1	6.42:1
mri8I16	8	67.82	0.0159	70.85	0.0118	19.80:1	8.95:1	6.17:1
mri8I17	8	68.66	0.0166	71.74	0.0122	19.90:1	9.18:1	6.28:1
mri8I18	8	67.07	0.0194	69.72	0.0149	19.70:1	9.12:1	6.23:1
mri8I19	8	67.08	0.0211	69.67	0.0163	19.70:1	8.82:1	6.09:1
mri8I20	8	67.10	0.0235	69.43	0.0186	19.85:1	9.72:1	6.52:1
mri8I21	8	68.20	0.0240	70.49	0.0191	21.29:1	11.82:1	7.60:1
mri8I22	8	67.83	0.0262	69.90	0.0213	22.22:1	11.48:1	7.57:1
mri8I23	8	67.69	0.0294	69.82	0.0237	23.09:1	11.41:1	7.63:1
Avg.		**68.82**	**0.0183**	**71.02**	**0.0147**	**20.90:1**	**10.42:1**	**6.95:1**

#QBits: Number of Quantization Bits for compressed patterns,
NE: Non-Edge, E+NE: Edge + Non-Edge (combined), PSNR: Peak Signal to Noise Ratio,
AMSE: Average Mean Squared Error, NE-CR: Non-Edge Compression Ratio (C.R.),
E-CR: Edge C.R., CCR: Combined (Edge+NonEdge images) C.R.

Table 7.4: Results of image compression using EPIC for Expt. CTEST4

EPIC Compression Expt. CTEST4_A								
Image	#QBits	NE-PSNR	NE-AMSE	E+NE-PSNR	E+NE-AMSE	NE-CR	E-CR	CCR
mri8I00	8	70.46	0.0269	72.64	0.0217	23.87:1	9.86:1	6.98:1
mri8I01	8	70.46	0.0181	72.49	0.0148	23.63:1	9.39:1	6.72:1
mri8I02	8	71.49	0.0142	73.35	0.0118	24.51:1	10.73:1	7.46:1
mri8I03	8	70.33	0.0144	71.99	0.0122	24.01:1	10.49:1	7.30:1
mri8I04	8	71.27	0.0129	73.04	0.0109	24.25:1	12.08:1	8.06:1
mri8I05	8	70.89	0.0127	72.66	0.0107	24.33:1	11.42:1	7.77:1
mri8I06	8	70.12	0.0169	72.01	0.0140	25.54:1	11.43:1	7.89:1
mri8I07	8	68.50	0.0200	70.54	0.0163	25.42:1	11.20:1	7.77:1
mri8I08	8	66.48	0.0269	68.47	0.0221	26.62:1	10.51:1	7.54:1
mri8I09	8	68.23	0.0193	70.16	0.0159	26.15:1	10.85:1	7.67:1
mri8I10	8	68.65	0.0154	70.48	0.0128	25.62:1	11.16:1	7.77:1
mri8I11	8	69.95	0.0119	71.69	0.0100	25.07:1	10.84:1	7.57:1
mri8I12	8	68.53	0.0122	70.38	0.0101	23.63:1	10.32:1	7.18:1
mri8I13	8	68.51	0.0120	70.98	0.0094	22.83:1	10.38:1	7.14:1
mri8I14	8	67.87	0.0144	70.58	0.0110	22.93:1	9.54:1	6.74:1
mri8I15	8	68.58	0.0152	71.49	0.0114	22.58:1	9.44:1	6.66:1
mri8I16	8	67.82	0.0159	70.85	0.0118	22.13:1	8.95:1	6.37:1
mri8I17	8	68.66	0.0166	71.74	0.0122	22.28:1	9.18:1	6.50:1
mri8I18	8	67.07	0.0194	69.72	0.0149	21.89:1	9.12:1	6.44:1
mri8I19	8	67.08	0.0211	69.68	0.0163	21.77:1	8.82:1	6.27:1
mri8I20	8	67.11	0.0234	69.44	0.0186	21.83:1	9.72:1	6.72:1
mri8I21	8	68.20	0.0240	70.49	0.0191	23.52:1	11.82:1	7.87:1
mri8I22	8	67.84	0.0262	69.90	0.0213	24.62:1	11.48:1	7.83:1
mri8I23	8	67.70	0.0293	69.84	0.0237	26.11:1	11.41:1	7.94:1
Avg.		**68.83**	**0.0183**	**71.02**	**0.0147**	**23.96:1**	**10.42:1**	**7.26:1**

#QBits: Number of Quantization Bits for compressed patterns,
NE: Non-Edge, E+NE: Edge + Non-Edge (combined), PSNR: Peak Signal to Noise Ratio,
AMSE: Average Mean Squared Error, NE-CR: Non-Edge Compression Ratio (C.R.),
E-CR: Edge C.R., CCR: Combined (Edge+NonEdge images) C.R.

Table 7.5: Results of image compression using EPIC for Expt. CTEST4_A

The same set of images were compressed using JPEG at compression ratios that were comparable to that achieved using EPIC. An image quality factor (QFactor) of 25 using JPEG provided a comparable average Compression Ratio to Expt. CTEST4 of 20.61:1, with an average AMSE of 0.0062, while a QFactor of 17 provided comparable average Compression Ratios to Expt. CTEST4_A. It was observed that as the compression ratios were increased (through lowering QFactor), the average AMSE also increased quite rapidly. This contrasts with the performance of EPIC, where the AMSE held constant as the compression ratio was increased. The results for JPEG for the case of QFactor = 25 and QFactor = 17 are presented in Table 7.6:

JPEG Compression Results (comparable compression ratios)								
Image	QFactor	PSNR	AMSE	CR	QFactor	PSNR	AMSE	CR
mri8I00	25	78.69	0.0118	20.21:1	17	75.56	0.0162	24.40:1
mri8I01	25	78.83	0.0078	19.56:1	17	75.57	0.0109	23.68:1
mri8I02	25	80.00	0.0061	19.95:1	17	76.64	0.0085	23.98:1
mri8I03	25	79.68	0.0056	19.63:1	17	76.37	0.0078	23.46:1
mri8I04	25	79.81	0.0055	19.89:1	17	76.64	0.0076	23.71:1
mri8I05	25	79.81	0.0052	20.20:1	17	76.53	0.0072	24.28:1
mri8I06	25	79.72	0.0065	20.91:1	17	76.36	0.0091	25.06:1
mri8I07	25	79.23	0.0068	20.73:1	17	76.03	0.0094	24.73:1
mri8I08	25	78.62	0.0080	20.73:1	17	75.43	0.0110	24.42:1
mri8I09	25	78.75	0.0067	20.94:1	17	75.63	0.0092	24.90:1
mri8I10	25	79.52	0.0052	21.01:1	17	76.37	0.0071	25.15:1
mri8I11	25	80.12	0.0043	21.28:1	17	76.90	0.0059	25.44:1
mri8I12	25	79.31	0.0041	20.16:1	17	76.12	0.0057	24.30:1
mri8I13	25	78.50	0.0044	19.73:1	17	75.32	0.0061	24.17:1
mri8I14	25	78.33	0.0051	19.90:1	17	75.21	0.0069	24.43:1
mri8I15	25	78.76	0.0055	20.77:1	17	75.79	0.0074	25.36:1
mri8I16	25	78.34	0.0056	20.18:1	17	75.39	0.0075	24.55:1
mri8I17	25	79.07	0.0059	20.55:1	17	75.99	0.0080	25.16:1
mri8I18	25	78.80	0.0060	19.78:1	17	75.76	0.0082	24.11:1
mri8I19	25	79.47	0.0061	20.43:1	17	76.44	0.0083	24.80:1
mri8I20	25	79.88	0.0065	20.21:1	17	76.73	0.0090	24.53:1
mri8I21	25	80.94	0.0067	21.75:1	17	77.74	0.0092	26.40:1
mri8I22	25	81.14	0.0069	22.41:1	17	77.98	0.0095	27.71:1
mri8I23	25	82.00	0.0070	23.77:1	17	78.73	0.0097	28.46:1
Avg.		**79.47**	**0.0062**	**20.61:1**		**76.30**	**0.0086**	**24.88:1**

QFactor: JPEG Quality Factor, PSNR: Peak Signal to Noise Ratio,
AMSE: Average Mean Squared Error, CR: Compression Ratio

Table 7.6: Results of image compression using JPEG with QFactor = 25 and 17

However, for error rates that were comparable to that achieved by EPIC, which corresponded to an average compression ratio of 32.15:1 and an average AMSE of 0.0132 using a QFactor of 10, the recovered JPEG images were much more distorted and “blocky”, and some features could not be distinguished. In contrast, recovered EPIC images indicated much better subjective image quality. The JPEG results for QFactor = 10 are presented in Table 7.7:

JPEG Compression Results (comparable error rates)				
Image	QFactor	PSNR	AMSE	CR
mri8I00	10	71.46	0.0244	32.08:1
mri8I01	10	71.29	0.0167	30.55:1
mri8I02	10	72.29	0.0131	30.93:1
mri8I03	10	71.99	0.0122	30.83:1
mri8I04	10	72.19	0.0118	30.83:1
mri8I05	10	72.22	0.0112	31.00:1
mri8I06	10	72.00	0.0140	31.80:1
mri8I07	10	71.49	0.0148	30.94:1
mri8I08	10	70.91	0.0173	30.62:1
mri8I09	10	71.10	0.0145	31.66:1
mri8I10	10	71.70	0.0113	32.27:1
mri8I11	10	72.63	0.0091	32.75:1
mri8I12	10	71.74	0.0088	31.61:1
mri8I13	10	71.10	0.0093	31.74:1
mri8I14	10	71.06	0.0105	31.80:1
mri8I15	10	71.70	0.0112	32.93:1
mri8I16	10	71.21	0.0114	32.09:1
mri8I17	10	71.80	0.0122	33.15:1
mri8I18	10	71.55	0.0124	31.89:1
mri8I19	10	71.93	0.0130	32.90:1
mri8I20	10	72.07	0.0143	32.08:1
mri8I21	10	73.30	0.0144	34.01:1
mri8I22	10	73.53	0.0148	34.77:1
mri8I23	10	74.06	0.0155	36.31:1
Avg.		**71.93**	**0.0132**	**32.15:1**

QFactor: JPEG Quality Factor, PSNR: Peak Signal to Noise Ratio,
AMSE: Average Mean Squared Error, CR: Compression Ratio

Table 7.7: Results of image compression using JPEG with QFactor = 10

7.3. Discussion of Results

Although the quantitative results from JPEG had better AMSE values at QFactor = 10, the subjective image quality of the recovered JPEG image was much worse than those obtained from EPIC which had comparable AMSE values. This was due to the fact that the neural network compressor distributed the errors much more evenly throughout the entire image. In addition, since JPEG is an algorithmic image compression scheme, there is no way to prevent the rapid deterioration in the quality of JPEG images at compression rates above 20:1, making JPEG impractical for applications requiring very high compression ratios. However, EPIC would be able to provide a means of achieving such high compression ratios with little or no deterioration in the subjective image quality through sufficiently trained networks.

The DANN neural network training scheme has very slow convergence properties. Consequently, the results detailed in the previous sections reflect the fact that some of the networks have not converged sufficiently to the desired error rates. Although the improvements proposed for EPIC helped circumvent some of its most obvious weaknesses, more work still remains to be done to improve the training and convergence characteristics. Some improvements suggested in [3] might help improve the convergence properties of the EPIC neural network compressors, thus enabling EPIC to achieve better error rates compared to JPEG while achieving much higher compression ratios. Another means of achieving better error rates is through the removal of edges from the image to be compressed by the EPIC neural network compressors. This edge-removed image would have less variability compared to the original image, thus making it easier for the neural network to process and compress the image, since neural network-based compression techniques tend to have low-pass properties and smooth out edge information.

Furthermore, the edge detection and preservation step incurs a rather substantial penalty in terms of the overall achievable compression ratios. Nonetheless, it is an important component of the compression scheme for meeting the criteria of a suitable compression system for MR images, as well as improve the overall image quality. As stated previously, edge information is also vital for 3D volume visualization and multiple modality image registration applications.

Examples of the recovered images for EPIC (Expt. CTEST4_A) and JPEG (QFactor = 10) are given below, in Figures 7.1 to 7.3:

Not Available in Softcopy Format

Figure 7.1: (left) Original, (middle) EPIC Expt. CTEST4_A, and (right) JPEG (Q=10) Recovered Images for MRI8I02

Not Available in Softcopy Format

Figure 7.2: (left) Original, (middle) EPIC Expt. CTEST4_A, and (right) JPEG (Q=10) Recovered Images for MRI8I11

Not Available in Softcopy Format

Figure 7.3: (left) Original, (middle) EPIC Expt. CTEST4_A, and (right) JPEG (Q=10) Recovered Images for MRI8I17

Chapter 8. Summary

8.1. Advantages of the EPIC Compression Scheme

(i) Edge preservation: Since the edges in the image are coded using a lossless scheme, it does not suffer from the drawback of conventional JPEG, which tends to lose high frequency components (edges) during compression.

(ii) Feature determination: A major advantage of the edge extraction step is that it provides the first step towards the extraction of features from the image for use in further image processing. This utilization of an essential step in image processing in the compression process helps eliminate redundant processing later on.

(iii) High compression ratio for non-edge images: the separation of edge information from the rest of the image affords the use of lossy compression schemes that achieve high compression ratios. In addition, the use of neural networks for performing this step enables us to tailor the compression architecture for specific types of images.

(iv) Progressive Transmission: Since neural network compressors are able to generalize and compensate for noisy inputs, it is therefore possible to provide progressive transmission of images by transmitting the most significant bits of the compressed vectors first. This is effectively similar to performing quantization on the transmitted vectors, where the received compressed vector is improved successively using more significant bits, increasing the number of quantization levels and converging to the final image.

(v) Temporal adaptability: Neural network based compressors are by definition adaptive. In addition, they can be configured to adapt to changes over time in the input data stream by the use of shadow neural network compressors which continue to learn from patterns extracted from the input stream. If the error rates of the online network exceed a given threshold, the shadow compressors can replace the existing network easily, thus achieving temporal adaptability.

8.2. Future Enhancements

Further improvements to the DANN training algorithms are still needed, to improve its speed of convergence and convergence ability beyond that achieved through the techniques proposed in EPIC. This is necessary due to the non-linear nature of the neural network training process, as well as the high degrees of freedom present in parameter selection. Edge removal prior to neural network compression would also help reduce the variability in the image and help convergence.

In order for the compression architecture to perform optimally, continuous monitoring of the compression process should be maintained. This is accomplished by using a similar bank of neural compressors in parallel with the actual compressor (called the shadow compressor), and if the error exceeds a predefined threshold, the shadow compressor network will be retrained in order to return the system to its operational goals. Once the training is completed, the updated weights of the neural networks will be transferred to the operational compressor network.

Variable sized image blocks can also be used to reduce “blocking” effects, while large image blocks can be used to provide even higher compression ratios for the uniform dark regions surrounding the MR images.

Appendix

A.1. Dynamic Autoassociative Neural Networks — Architecture & Training

The classic multilayer feed forward network architecture for data compression proposed by Cottrel, Munro, and Zipser [23] comprises of a three layer network, with N neurons each in the input and output layers forming an autoassociative mapping, and M neurons in the hidden layer ($M < N$) to represent the compressed version of the data. The output values from the hidden layer neurons are used to reconstruct the data set. It has been shown by Baum [34] and Widrow [35] that the capacity or learning ability of a network is dependent on the number of weight connections. With a three layer network, a fixed compression ratio dictates the capacity of the network for storing and reconstructing different input patterns, and vice-versa. This severely limits the usefulness of neural network based data compression systems. In order to achieve very high compression ratios while adapting to different data characteristics, four layer feedforward neural networks are therefore used to provide the necessary learning capacity and adaptability. In order to correctly learn complex datasets, the number of neurons in the first hidden layer is variable and can be tailored to the complexity of the data, whereas the second hidden layer provides the desired compression ratio. This is illustrated in Figure A.1.

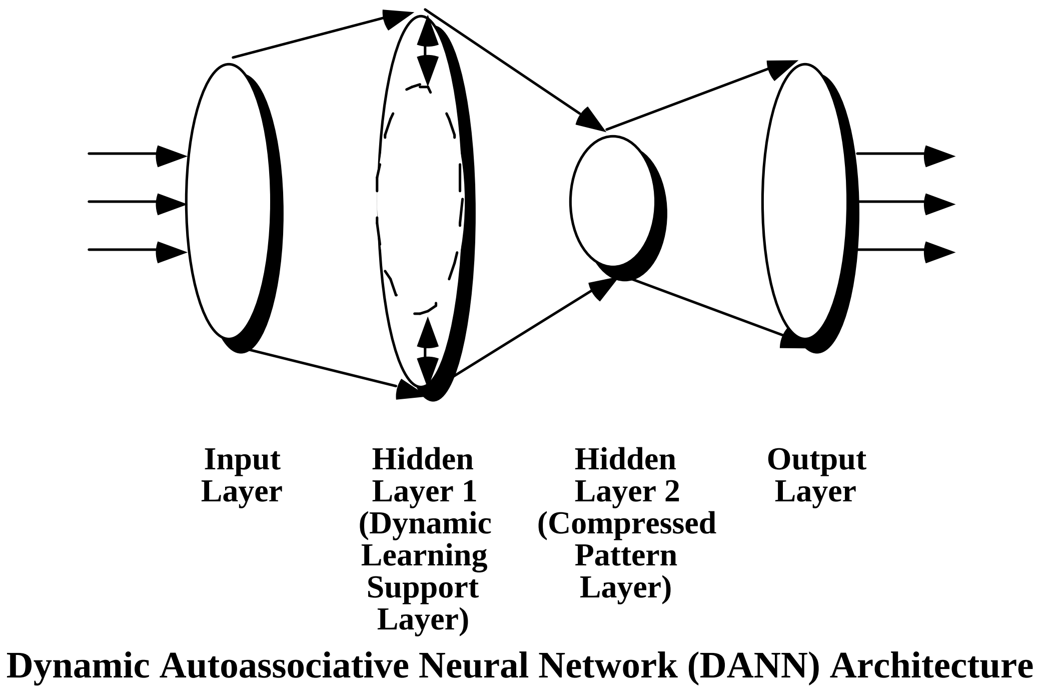

Input Layer
Hidden Layer 1 (Dynamic Learning Support Layer)
Hidden Layer 2 (Compressed Pattern Layer)
Output Layer
Dynamic Autoassociative Neural Network (DANN) Architecture

Figure A.1: DANN Architecture

However, the use of four layer networks pose other problems. It was reported in [36] that although three and four layer networks with similar number of weights give statistically identical results (i.e., they are equivalent), four layer networks with uneven numbers of weights in the two hidden layers (unbalanced networks) are difficult to train and often result in the error function being trapped in a local minima. In addition, the need to correctly learn datasets of arbitrary complexity results in costly architectural modification and retraining on a trial and error basis using conventional statically defined network architectures. In order to address the difficulty of learning for unbalanced networks, the descending epsilon technique proposed by Yu and Simmons [27] is used during training. The error present at each output neuron obtained during training is compared to a threshold epsion (ε), and if it is less than ε, the error is ignored. Otherwise, that error is backpropagated as usual to the hidden layer neurons. Once all the training patterns pass a given ε, it is lowered and training continues until a final ε is reached. A very similar approach to descending epsilon has been used successfully for reduced precision neural networks in [37], where it is called the Weighted Error Function. The use of the descending epsilon technique also constrains the errors into a more well defined range, so that individual pattern errors are well constrained, in addition to the average error for the network across all patterns.

To provide for adaptation to arbitrary datasets during training, and to avoid the expense of a trial and error approach to network architecture determination, dynamic neural network architectures have been proposed [38], and more recently [39], used dynamic network architectures to allow the system to escape local minima and provide necessary learning support. [39] further indicates that better convergence properties are observed when nodes are added to the first hidden layer compared to the second hidden layer. This is due to the fact that the second hidden layer acts as a “filter” to buffer the changes in the network architecture due to the addition of new neurons. Since the requirements of the data compression problem dictate that network modifications can occur only in the first hidden layer, the results presented in [39] further validate this approach. These techniques have been used to create the DANN compression system [2]. It combines the dynamic architecture approach with the use of the descending epsilon technique during training to achieve low errors with high compression ratios.

During the training phase for the neural networks, training images are first segmented into smaller subimages and converted into training patterns. Training sets are tailored to the specific type of images and image modality in order to achieve optimal performance for a given modality. The Mean Squared Error (MSE) of the neural networks output is used to quantify the performance of the training process. However, it must be noted that MSE alone is not sufficient for determining whether the network is converging to the desired visual output. A subjective criteria based on the visual fidelity of the image is desirable in order to provide a more balanced approach to training the neural networks.

A.2. Backpropagation Formulae

The formulae used for the four-layered feedforward perceptron network, as adapted from the formulae given in [40] are:

Input layer, given p-th input vector, layer size N:

$$x_p = \left(x_{p1}, x_{p2}, \ldots, x_{pN}\right)^t$$

First hidden layer input for j-th neuron with bias unit, layer size N_{h1}:

$$net_{pj}^{h1} = \sum_{i=1}^{N} w_{ji}^{h1} x_{pi} + \theta_j^{h1}$$

First hidden layer output for j-th neuron:

$$i_{pj}^{h1} = f_j^{h1}\left(net_{pj}^{h1}\right)$$

Second hidden layer input for k-th neuron with bias unit, layer size N_{h2}:

$$net_{pk}^{h2} = \sum_{j=1}^{N_{h1}} w_{kj}^{h2} i_{pj}^{h1} + \theta_k^{h2}$$

Second hidden layer output for k-th neuron:

$$i_{pk}^{h2} = f_k^{h2}\left(net_{pk}^{h2}\right)$$

Output layer input for l-th neuron, layer size N:

$$net_{pl}^{o} = \sum_{k=1}^{N_{h2}} w_{lk}^{o} i_{pk}^{h2} + \theta_l^{o}$$

Output layer output for l-th neuron:

$$o_{pl} = f_l^{o}\left(net_{pl}^{o}\right)$$

Sigmoidal function of the outputs from each neuron and its derivative:

$$f_y^x\left(net_{py}^x\right) = \left(1 + e^{-net_{py}^x}\right)^{-1}$$

$$f_y^{x'}\left(net_{py}^x\right) = f_y^x\left(net_{py}^x\right)\left[1 - f_y^x\left(net_{py}^x\right)\right]$$

Error terms for the output units:

$$\delta_{pl}^{o} = \left(y_{pl} - o_{pl}\right) f_{l}^{o'}\left(net_{pl}^{o}\right)$$

Error terms for the second hidden layer units:

$$\delta_{pk}^{h2} = f_{k}^{h2'}\left(net_{pk}^{h2}\right)\sum_{l=1}^{N} \delta_{pl}^{o} w_{lk}^{o}$$

Error terms for the first hidden layer units:

$$\delta_{pj}^{h1} = f_{j}^{h1'}\left(net_{pj}^{h1}\right)\sum_{k=1}^{N_{h2}} \delta_{pk}^{h2} w_{kj}^{h2}$$

Weight updates for the output layer:

$$w_{lk}^{o}(t+1) = w_{lk}^{o}(t) + \eta \delta_{pl}^{o} i_{pk}^{h2} + \alpha \Delta_{p} w_{lk}^{o}(t-1)$$

Weight updates for the second hidden layer:

$$w_{kj}^{h2}(t+1) = w_{kj}^{h2}(t) + \eta \delta_{pk}^{h2} i_{pj}^{h1} + \alpha \Delta_{p} w_{kj}^{h2}(t-1)$$

Weight updates for the first hidden layer:

$$w_{ji}^{h1}(t+1) = w_{ji}^{h1}(t) + \eta \delta_{pj}^{h1} x_{pi} + \alpha \Delta_{p} w_{ji}^{h1}(t-1)$$

The error term for pattern p:

$$E_{p} = \frac{1}{2}\sum_{l=1}^{N} \delta_{pl}^{o\,2}$$

References

[1] Clarke, R., *Transform Coding of Images,* pp. 405-406, London: Academic Press, Harcourt Brace Jovanovich, Publishers, 1985.

[2] Rios, A., Kabuka, M., "Image Compression with a Dynamic Autoassociative Neural Network," *Proceedings of ANNIE '92*, pp. 503–510, St. Louis, MO, Nov. 1992.

[3] Rios, A., Kabuka, M., "A High Performance Compression System: An Application of Generative Neural Network Algorithms," *Proceedings of the Data Compression Conference 1993*, Salt Lake City, UT, March 1993.

[4] Hamming, R., *Coding and Information Theory*, 2nd Ed., Englewood Cliffs, NJ: Prentice Hall, 1986.

[5] Rabbani, M., Jones, P., *Digital Image Compression Techniques*, Vol. TT7, SPIE Optical Engineering Press, 1991.

[6] Glenn, M., "Image Compression for Medical Imaging Systems," *Journal of Medical Systems*, Vol. 11, Nos. 2/3, pp. 149–156, 1987.

[7] Akisada, M., "Present Status and Perspective of PACS Activities in Japan," *Proceedings on the First International Conference on Image Management and Communications in Patient Care: Implementation and Impact,* pp. 25–31, Washington, D.C., 1989.

[8] Netravali, A., Limb, J., "Picture Coding: A Review," *Proceedings of the IEEE,* Vol. 68, No. 3, pp. 366–406, March 1980.

[9] Jain, A., "Image Data Compression: A Review," *Proceedings of the IEEE,* Vol. 69, No. 3, pp. 349–389, March 1981.

[10] Bramble, J., Cook, L., et. al., "Image Data Compression in Magnification Hand Radiographs," *Radiology,* Vol. 170, No. 1, pp. 133–136, Jan. 1989.

[11] Howe, F., "Implementation and Evaluation of Data Compression of MR Images," *Magnetic Resonance Imaging,* Vol. 7, No. 2, pp. 127–132, 1989.

[12] Ishigaki, T., Sakuma, S., et. al., "Clinical Evaluation of Irreversible Image Compression: Analysis of Chest Imaging with Computed Radiography," *Radiology*, Vol. 175, No. 3, pp. 739–743, June 1990.

[13] Chan, K., Lou, S., Huang, H., "Radiological Image Compression Using Full-Frame Cosine Transform with Adaptive Bit-Allocation," *Computerized Medical Imaging and Graphics*, Vol. 13, No. 2, pp. 153-159, 1989.

[14] Chen, W., Smith, C., "Adaptive Coding of Monochrome and Color Images," *IEEE Trans. on Communications,* Vol. COM-25, No. 11, pp. 1285-1292, Nov. 1977.

[15] Riskin, E., Lookabaugh, T., et. al., “Variable Rate Vector Quantization for Medical Image Compression,” *IEEE Transactions on Medical Imaging*, Vol. 9, No. 3, pp. 290–298, Sept. 1990.

[16] Markas, T., Reif, J., “Image Compression Methods with Distortion Controlled Capabilities,” *Proceedings of the Digital Compression Conference 1991*, 1991.

[17] Safranek, R., Johnston, J., “A Perceptually Tuned Sub-band Image Coder With Image Dependent Quantization and Post-quantization Data Compression,” *ICASSP*, pp. 1945–1948, 1989.

[18] Halpern, E., et. al., “Quadtree-Based Data Compression of Abdominal CT Images,” *Investigative Radiology*, Vol. 25, No. 1, pp. 31–38, Jan. 1990.

[19] Halpern, E., et. al., “Application of Region of Interest Definition to Quadtree-Based Compression of CT Images,” *Investigative Radiology*, Vol. 25, No. 6, pp. 703–707, June 1990.

[20] Wu, Y., Coll, D., “BTC-VQ-DCT Hybrid Coding of Digital Images,” *IEEE Transactions on Communications*, Vol. COM-39, No. 9, pp. 1283-1287, Sept. 1991.

[21] Hecht-Nielsen, R., *Neurocomputing*, Reading, MA: Addison-Wesley, 1989/1990.

[22] Fang, W., Sheu, B., Chen, O., “A Real-Time VLSI Neuroprocessor for Adaptive Image Compression Based Upon Frequency-Sensitive Competitive Learning,” *Proceedings of the International Joint Conference on Neural Networks*, Vol. I, pp. 429-435, Seattle, WA, 1991.

[23] Cottrell, G., Munro, P., Zipser, D., “Image Compression by Back Propagation: An Example of Extensional Programming,” *Advances in Cognitive Science*, pp. 209-240, Ed. Sharkey, N., Norwood, NJ: Ablex, 1989.

[24] Sonehara, N., Kawato, M., et. al., “Image Data Compression Using a Neural Network Model,” *Proceedings of the International Joint Conference on Neural Networks*, Vol. II, pp. 35–41, Washington D. C., June 1989.

[25] Arozullah, M., Namphol, A., “A Data Compression System using Neural Network Based Architecture,” *Proceedings of the International Joint Conference on Neural Networks*, Vol I, pp. 531–536, San Diego, CA, 1990.

[26] Mougeot, M., Azencott, R., Angeniol, B., “Image Compression with Back Propagation: Improvement of the Visual Restoration Using Different Cost Functions,” *Neural Networks*, Vol. 4, pp. 467–476, 1991.

[27] Yu, Y., Simmons, R., “Descending Epsilon in Back Propagation: A Technique for Better Generalization,” *Proceedings of the International Joint Conference on Neural Networks*, Vol III, pp. 167–172, San Diego, CA, 1990.

[28] Garrido, D., Nunes, R., "An Edge/Non-Edge Image Compression Algorithm," *Proceedings of the IEEE International Conference On Image Processing*, ICIP '89 Vol. 2, pp. 519–523, Singapore, Sept. 1989.

[29] Bidaut, L., "Composite PET and MRI for accurate localization and metabolic modeling: a very useful tool for research and clinic," *Proceedings of the SPIE, Medical Imaging V: Image Processing,* Vol. 1445, pp. 66–77, Feb. 1991.

[30] Chen, C., Ouyang, X., et. al., "Incorporation of structural CT and MR images in PET image reconstruction," *Proceedings of the SPIE, Medical Imaging V: Image Processing*, Vol. 1445, pp. 222–225, Feb. 1991.

[31] Gee, J., Reivich, M., et. al., "Evaluation of multiresolution elastic matching using MRI data," *Proceedings of the SPIE, Medical Imaging V: Image Processing*, Vol. 1445, pp. 226–234, Feb. 1991.

[32] Castan, S., Zhao, J., Shen, J., "New Edge Detection Methods Based on Exponential Filter," *Proceedings 10th Intl. Conf. on Pattern Recognition*, Vol. 1, pp. 709–711, Atlantic City, New Jersey, June 1990.

[33] Canny, J., "A Computational Approach to Edge Detection," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. PAMI-8, No. 6, pp. 679–698, Nov. 1986.

[34] Baum, E., Haussler, D., "What Size of Net Gives Valid Generalization?" *Neural Computation*, Vol. 1, No. 1, Spring 1989.

[35] Widrow, B., "30 years of Adaptive Neural Networks: Perceptron, Madaline and Backpropagation," *Proceedings of IEEE*, Vol. 78, No. 9, pp. 1415–1442, 1990.

[36] de Villiers, J., Barnard, E., "Backpropagation Neural Nets with One and Two Hidden Layers," *IEEE Transactions on Neural Networks*, Vol. 4, No. 1, pp. 136–141, Jan. 1993.

[37] Sakaue, S., Kohda, T., et. al., "Reduction of Required Precision Bits for Back-Propagation Applied to Pattern Recognition," *IEEE Transactions on Neural Networks,* Vol. 4, No. 2, pp. 242–256, March 1993.

[38] Hirose, Y., Ymamshita, K., Hijiya, S., "Backpropagation Algorithm Which Varies the Number of Hidden Units," *Neural Networks*, Vol. 4, No. 1, pp. 61–66, 1991.

[39] Azimi-Sadjadi, M., Sheedvash, S., Trujillo, F., "Recursive Dynamic Node Creation in Multilayer Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 4, No. 2, pp. 242–256, March 1993.

[40] Freeman, J., Skapura, D., *Neural Networks: Algorithms, Applications and Programming Techniques,* Cambridge, MA: Addison Wesley, 1991.