9-1-1996

IMAGE COMPRESSION AND SIGNAL CLASSIFICATION BY NEURAL NETWORKS AND PROJECTION PURSUITs

M. Fardanesh
Purdue University School of Electrical and Computer Engineering

O. K. Ersoy
Purdue University School of Electrical and Computer Engineering

Follow this and additional works at: http://docs.lib.purdue.edu/ecetr

http://docs.lib.purdue.edu/ecetr/93

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
IMAGE COMPRESSION AND SIGNAL CLASSIFICATION BY NEURAL NETWORKS AND PROJECTION PURSUITS

M. FARDANESH
O. K. ERSOY

TR-ECE 96-15
SEPTEMBER 1996

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, INDIANA 47907-1285
IMAGE COMPRESSION AND SIGNAL CLASSIFICATION BY NEURAL NETWORKS AND PROJECTION PURSUITS

M. Fardanesh, O. K. Ersoy

Purdue University
School of Electrical and Computer Engineering
W. Lafayette, IN 47906-1285
TABLE OF CONTENTS

| LIST OF TABLES | vi |
| LIST OF FIGURES | vii |
| ABSTRACT | x |
| CHAPTER 1 - INTRODUCTION | 1 |
| 1.1 Introduction | 1 |
| 1.2 The backpropagation algorithm | 5 |
| 1.3 Organization of thesis | 7 |
| CHAPTER 2 - MODEL BASED IMAGE COMPRESSION USING NEURAL NETWORKS | 9 |
| 2.1 Introduction | 9 |
| 2.2 JPEG baseline system | 13 |
| 2.3 Segmentation | 17 |
| 2.4 Image modeling by a neural network | 22 |
| 2.5 Quantization | 24 |
| 2.6 Arithmetic coding | 25 |
| 2.7 Experimental results | 31 |
| 2.8 Conclusions | 34 |
| CHAPTER 3 - IMAGE CODING WITH NEURAL NETWORKS AND PROJECTION PURSUITS | 39 |
| 3.1 Introduction | 39 |
| 3.2 Quadtrees segmentation | 40 |
| 3.3 Image coding with neural networks and projection pursuits | 45 |
3.4 Steepest descent algorithm................................................................. 49
3.5 The reduced gradient method............................................................ 50
3.6 Experimental results......................................................................... 54
3.7 Conclusions....................................................................................... 62

CHAPTER 4. COLOR IMAGE CODING BY NEURAL NETWORKS AND
PROJECTION PURSUITS............................................................................ 71
4.1 Introduction......................................................................................... 71
4.2 Human visual system.......................................................................... 72
4.3 NTSC color transformation................................................................. 74
  4.3.1 Color coordinate systems and transforms...................................... 74
  4.3.2 The NTSC color transformation.................................................... 79
4.4 Quadtree segmentation of color images........................................... 82
4.5 Color image coding by neural networks and projection pursuits......... 86
4.6 Experimental results.......................................................................... 87
4.7 Conclusions....................................................................................... 92

CHAPTER 5. CLASSIFICATION ACCURACY IMPROVEMENT OF NEURAL
NETWORK CLASSIFIERS BY USING UNLABELED DATA... 97
5.1 Introduction......................................................................................... 97
5.2 Neural network classifiers................................................................... 100
5.3 The backpropagation algorithm......................................................... 102
5.4 Effect of additional unlabeled observations....................................... 102
5.5 The Colorado data set.......................................................................... 106
5.2 Experimental Results.......................................................................... 108
5.2 Conclusions....................................................................................... 118

CHAPTER 6. SUMMARY AND CONCLUDING REMARKS ................... 119
6.1 Summary of results............................................................................ 119
6.2 Future research.................................................................................. 121
LIST OF REFERENCES
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>AC coefficient grouping</td>
</tr>
<tr>
<td>2.2</td>
<td>The fixed probability model for alphabet {a, e, i, o, u, !}</td>
</tr>
<tr>
<td>2.3</td>
<td>The number of segmented blocks produced by quadtree segmentation</td>
</tr>
<tr>
<td>2.4</td>
<td>The number of fixed size blocks as a function of number of hidden layer nodes</td>
</tr>
<tr>
<td>4.1</td>
<td>The number of blocks resulted from quadtree segmentation of images Y, I, and Q for different block sizes for the test image Lenna</td>
</tr>
<tr>
<td>4.2</td>
<td>The bit-rate of the encoded test image Lenna with the two algorithms, PSNR based algorithm, and JPEG</td>
</tr>
<tr>
<td>5.1</td>
<td>Training and testing samples for each class on Colorado data</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>The activation functions. (a) log-sigmoid function. (b) tan-sigmoid function</td>
<td>7</td>
</tr>
<tr>
<td>2.1</td>
<td>DCT based encoder.</td>
<td>14</td>
</tr>
<tr>
<td>2.2</td>
<td>DCT based decoder.</td>
<td>14</td>
</tr>
<tr>
<td>2.3</td>
<td>The structure of quadtree segmentations. a) original image. b) top-down quadtree. c) bottom-up quadtree.</td>
<td>19</td>
</tr>
<tr>
<td>2.4</td>
<td>The tree structure of quadtree segmentations of the final block.</td>
<td>20</td>
</tr>
<tr>
<td>2.5</td>
<td>The variance based quadtree segmentation of the test image Lenna.</td>
<td>21</td>
</tr>
<tr>
<td>2.6</td>
<td>Neural network implementation of the proposed algorithm.</td>
<td>23</td>
</tr>
<tr>
<td>2.7</td>
<td>Representation of the arithmetic coding process.</td>
<td>30</td>
</tr>
<tr>
<td>2.8</td>
<td>The test image Lenna.</td>
<td>36</td>
</tr>
<tr>
<td>2.9</td>
<td>The reconstructed test image Lenna with the proposed algorithm.</td>
<td>37</td>
</tr>
<tr>
<td>2.10</td>
<td>The JPEG decoded image with a bit rate of 2.5 bpp and PSNR of 30 dB.</td>
<td>38</td>
</tr>
<tr>
<td>3.1</td>
<td>The variance based quadtree segmentation of the test image Pepper.</td>
<td>42</td>
</tr>
<tr>
<td>3.2</td>
<td>The PSNR based quadtree segmentation of the test image Lenna.</td>
<td>43</td>
</tr>
<tr>
<td>3.3</td>
<td>The PSNR based quadtree segmentation of the test image Pepper.</td>
<td>44</td>
</tr>
<tr>
<td>3.4</td>
<td>Neural network implementation of the proposed algorithm.</td>
<td>48</td>
</tr>
<tr>
<td>3.5</td>
<td>The test image Lenna.</td>
<td>63</td>
</tr>
<tr>
<td>3.6</td>
<td>The test image Pepper.</td>
<td>63</td>
</tr>
</tbody>
</table>
3.7 The image coding procedure with PSNR based \textit{quadtree} segmentation, neural networks, and projection pursuits .................................................. 60

3.8 The image coding procedure with variance based \textit{quadtree} segmentation, neural networks, and projection pursuits ........................................ 61

3.9 The encoded test image Lenna with variance based \textit{quadtree segmentation} 65

3.10 The encoded test image Lenna with PSNR based \textit{quadtree} segmentation 66

3.11 The PEG encoded test image Lenna at bit-rate of 14 bpp and

PSNR of 27 dB............................................................................................................. 67

3.12 The encoded test image Pepper with variance based \textit{quadtree} segmentation 68

3.13 The encoded test image Pepper with PSNR based \textit{quadtree} segmentation 69

3.14 The PEG encoded test image Pepper at bit-rate of 14 bpp and

PSNR of 21.62 dB........................................................................................................... 70

4.1 \textit{Quadtree} segmentation map for the 512x512 luminance image Lenna....... 83

4.2 \textit{Quadtree} segmentation map for the 256x256 chrominance image I for Lenna 84

4.3 \textit{Quadtree} segmentation map for the 256x256 chrominance image Q for Lenna. 85

4.4 The color image coding procedure with NTSC color \textit{transformation}, PSNR based \textit{quadtree} segmentation, neural networks, and projection \textit{pursuits}............. 91

5.1 Model of neuron........................................................................................................ 101

5.2 Structure of the three-layer neural network with ten neurons

in the output layer. 101

5.3 Supervised-unsupervised learning procedure....................................................... 109

5.4 Classification accuracy of neural network classifier for each class (a) class 1, (b) class 2, (c) class 3, (d) class 4, (e) class 5, (f) class 6, (g) class 7, (h) class 8, (i) class 9, (j) class 10......................................................................................... 112
5.5 Classification accuracy of the neural network with supervised-unsupervised learning for a typical seed

5.6 The sum squared error of the neural network classifier with Supervised-unsupervised learning for a typical seed

5.7 Average Classification accuracy of the neural network with supervised-unsupervised learning for three different seeds

5.8 Average the sum squared error of the neural network classifier with Supervised-unsupervised learning for three different seeds
ABSTRACT

In this report, two applications of neural networks are investigated. The first one is low bit rate image compression by using neural networks and projection pursuit. The second one is improving the classification accuracy of neural network classifiers by using unlabeled data.

In the first part, a novel approach for low bit rate image coding is presented. The image is compressed by first \textit{quadtree} segmenting the image into blocks of different sizes based on two activity measures, and then constructing a distinct code for each block by \textit{invoking} the theory of projection pursuit. The two activity measures used in this work are the block variance and the signal to noise ratio (PSNR) of the reconstructed block. It is shown that the projection pursuit coding algorithm can adaptively construct a better \textit{approximation} for each block until the desired signal to noise ratio or bit rate is achieved. This method also adaptively finds the optimum network configuration. Experimental values for the objective measure of performance using PSNR are superior to the JPEG decoded images. The subjective quality of the encoded images with the proposed algorithm are also superior to the JPEG encoded images.

In the second part, classification accuracy improvement of neural network classifiers using unlabeled testing data is presented. In order to fully utilize the information contained in high dimensional data, training samples are needed from all classes. In order to \textit{increase} classification accuracy without increasing the number of training samples, the network makes use of testing data along with training data for learning. However, the testing data are unlabeled whereas the training data are labeled. It was \textit{shown} previously for the case of parametric classifiers that decision rules which use both labeled (training) and unlabeled (testing) samples have a lower expected error than those which use labeled samples only. Since the output of a neural network such as backpropagation network approximates the a posteriori probabilities, the same result applies to neural network classifiers. It is shown that including unlabeled samples from under-represented classes
in the training set improves the classification accuracy of some of the classes during supervised-unsupervised learning.
CHAPTER 1. INTRODUCTION

1.1. Introduction

Recent interest in artificial neural networks has motivated a large number of applications covering a wide range of research fields. The ability of learning in neural networks provides an interesting alternative to other conventional research methods. In this thesis, two applications of neural networks is investigated. The first one is the model based image compression by using neural networks and projection pursuits. The second one is the classification accuracy improvement of neural network classifiers using unlabeled data. The two applications demonstrate that neural networks can be effectively used for signal/image modeling as well as signal/image classification.

Visual communication is a rapidly evolving field for telecommunications, computer and media industries. Recent progress in electronics technology and broadband communication networks are leading to an increasing interest in a variety of applications such as video telephone, medical imaging, high-definition television (HDTV), remote surveillance, education, video mail, and entertainment [24]. Essential for the introduction of new communication services is low cost. The classical tradeoff between the required visual quality and the allowed bandwidth can not be solved without image compression. Bit rate reduction can be achieved by source coding schemes such as predictive coding, transform coding, subband coding and vector quantization [8]. The common characteristic of these compression algorithms is to exploit redundancy in digital images.
In other words, the goal is to achieve the best image fidelity for an available communication bit rate capacity.

To facilitate worldwide interchange of digitally encoded audiovisual data across many diverse applications, there is a demand for international standards for the coding methods and transmission formats. The existence of a standard is often the trigger to volume production of integrated circuits (VLSI), and it can significantly reduce the cost of specialized hardware required in many real-time data communication systems. To accomplish this task, the International Standards Organization (ISO) has introduced several compression standards. The Joint Photographic Expert Group (JPEG) of ISO has specified an algorithm for compression of still images [40]. This algorithm is based on fixed segmentation blocks and transform coding. The Consultative Committee of the International Telephone and Telegraph (CCITT) proposed the H.261 standards for video telephony and video conferencing [27]. The Motion Picture Experts Group (MPEG) proposed the MPEG-1 standard which is used for interactive video and provides picture quality comparable to VCR quality at rates of 1.5 Mbs [23]. MPEG made substantial progress in its later version MPEG-2 which will provide audiovisual quality of both broadcast TV and HDTV [1].

Digital images can usually be divided into different size blocks with variable amounts of detail and information. Fixed size, fixed shape segmentation has the advantage of simplicity and lower overhead, but usually fails to capture the information regarding the homogeneity of the region. Some popular segmentation algorithms are region growing and quadtree [21], [31]. The quadtree segmentation algorithm is particularly attractive because it can provide the flexibility of a variable blocksize partitioning while avoiding excessive overhead or side information by restricting the shape and the number of
possible sizes of the final regions from a predetermined set of options. The quadtree based segmentation encoder can effectively allocate the bits between the variable sized-blocks with different amount of activities.

Digital color images are usually represented by three bands, corresponding to red, green, and blue tristimulus values. Since there is often substantial correlation between the color planes, independent processing of each plane is not optimal. The traditional approach in color image processing is to apply a coordinate transformation to the RGB image to obtain the luminance and chrominance components [8]. A luminance-chrominance coordinate space is preferred because the image components are then relatively uncorrelated. The chrominance images usually have less spatial bandwidth than the luminance image. Therefore, the chrominance images are usually subsampled by two both horizontally and vertically. Monochrome image processing techniques can then be applied to the luminance and the subsampled chrominance images. Finally, an inverse coordinate transformation is applied to reconstruct the image.

The overall goal of any image compression technique is to represent an image with the smallest possible number of bits. So the objective is to optimize the classical tradeoff between the amount of compression and the quality of the reconstructed image. Current major trends in image compression includes the traditional transform based methods, vector quantization, and more recent subband coding approaches [8], [43], [15]. The recent interest in artificial neural networks has motivated a large number of applications in image compression. The efforts have mostly concentrated in artificial neural network implementation of vector quantization and autoassociate memory representation [10], [5]. In this thesis (chapters two, three, and four), a new neural network compression algorithm is presented which is based on the projection pursuit algorithm. In this new technique,
Image compression is obtained by segmenting an image into regions of different sizes, based on perceptual variation or signal to noise ratio of the reconstructed region and then constructing a distinct code for each region by invoking the theory of projection pursuits. Neural network implementation of this algorithm allows one to adaptively find the optimum number of parameters and the parameter values which is required to code a region. The process is terminated when the desired signal to noise ratio or bit rate is achieved.

Recently, there has been considerable amount of interest in using artificial neural networks to classify remote sensing data [3], [4], [38]. The ability of learning in neural networks provides an interesting alternative to the conventional classification methods. In remote sensing data classification, neural network models do not require any a priori knowledge of the class statistical distribution in data sources. In other words, they are distribution free. They also avoid the problem in statistical multisource analysis of specifying how much influence each source should have on classification. In chapter five, the classification accuracy improvement of neural network classifiers using unlabeled data is presented. In order to design an accurate classifier, training samples should represent all classes of interest. In remote sensing applications, the number of training samples which can be obtained are usually limited. Moreover, in some cases, the training samples may not be a good representation of the entire class, especially when the samples are collected from spatially adjacent regions. One approach to ease the problem is to include the unlabeled observations in the process of designing neural network classifiers. In summary, the purpose of this work is to explore and study some techniques for improving the classification accuracy of neural network classifiers by utilizing unclassified observation that may be available in large numbers and with no extra cost.
1.2. The Backpropagation Algorithm

In this thesis, multilayer perceptrons have been applied successfully to image compression and data classification by training them in a supervised manner with a highly popular algorithm known as the error back propagation algorithm. In early 1960's, Widrow and Hoff developed the delta rule [16]. It is a supervised learning approach for a two-layer neural network with linear activation functions. This learning rule can be used to calculate changes in the weights and biases so as to minimize the sum squared error of the network. The delta rule uses the method of steepest descent to minimize square error. The delta rule can be implemented to discriminate linearly separable data.

Backpropagation was created by generalizing the delta rule to multiple layer networks and nonlinear differentiable transfer functions [16]. It can be used to discriminate data that are not linearly separable. As in the delta rule, backpropagation involves adjusting the weights and biases of the network so as to minimize the sum squared error of the network. This is done by changing the values of the network weights and biases in the direction of steepest descent with respect to error. Backpropagation training may lead to a local minimum rather than a global one. In some cases the obtained local minimum may be satisfactory, but if it is not, a network with a different structure or different initial weights can be tried.

The backpropagation algorithm [16] which is used in this thesis can be summarized in the following steps.

Step 1: Initialize all the weights and biases to a small number between -1 and +1.
Step 2: Read the input vector, x and desired output vector, d.
step 3: Compute the actual outputs of the network defined as

\[ y_j = f\left(\sum_i w_{ij}x_i + q_j\right) \]

where the function \( f(\bullet) \) is the nonlinear activation function.

Step 4: Adjust the weights by

\[ w_{ij}(t + 1) = w_{ij}(t) + \eta \delta_j y_i + \alpha (w_{ij}(t) - w_{ij}(t - 1)) \]

where \( w_{ij}(t) \) is the weight from node \( i \) to node \( j \) at time \( t \), \( \eta \) is the learning rate, \( \alpha \) is a positive number between zero and one, and \( \delta_j \) is an error term for node \( j \). If node \( k \) is an output node, then

\[ \delta_k = f'(y_k)(d_k - y_k) \]

If the node \( j \) is an internal hidden node, then

\[ \delta_j = f'(y_k) \sum_k \delta_k w_{jk} \]

Step 5: Compute the mean squared error between the desired output and the actual output denoted by \( E \) as

\[ E = \frac{1}{2} \sum_j (d_j - y_j)^2 \]

If \( E \) becomes smaller than some predefined error goal, then stop the iterations, otherwise go to step 2.
The transfer functions which are used in this work are log-sigmoid and tan-sigmoid. The log-sigmoid and tan-sigmoid functions map a neuron input from the interval of \((-\infty, \infty)\) into the interval of \((0, 1)\) and \((-1, 1)\) respectively. These functions are differentiable and can be used in the backpropagation learning algorithm. The graph of log-sigmoid and tan-sigmoid is shown in Figure 1.1.

![Activation Functions](image)

Figure 1.1. The activation functions, (a) log-sigmoid function, (b) tan-sigmoid function

1.3. Organization Of Thesis

The thesis consists of six chapters. In chapter two, a novel approach for compression of grayscale images is presented. In this algorithm, central to our coding approach is the use of the variable-rate coding that conceptually corresponds to the idea of designing neural networks with different number of hidden layers. In chapter three, a new variable-rate coding technique is introduced which is based on the implementation of the projection pursuits algorithm. This method finds adaptively the optimum network; configuration and
is found superior to the proposed method in chapter two. In chapter four, model based image compression using neural network and projection pursuit is extended to color images. It is shown that the proposed model based image compression algorithm can effectively encode the luminance and subsampled chrominance images.

In chapter five, the classification accuracy improvement of neural network classifiers using unlabeled data is presented. It is shown that including unlabeled samples in the learning process can reduce the mean squared error and therefore, increase the classification accuracy of the neural network classifiers. Chapter six consists of conclusions and discussion of potential future research topics.
CHAPTER 2: MODEL BASED IMAGE COMPRESSION USING NEURAL NETWORKS

2.1. Introduction

A common characteristic of images is that neighboring pixels are highly correlated. To represent the image directly in terms of the pixel values is therefore insufficient. The art of image and data compression is concerned with minimizing the number of bits required to represent an image. Applications of data compression are primarily in transmission and storage of information. Image transmission applications are in broadcast television, teleconferencing, computer communications, facsimile transmission, radar and sonar. Image storage is required for educational and business documents, medical images, motion pictures, satellite images, and weather maps [26]. Image data compression methods can be classified in two basically different categories. In the first category are those methods which exploit redundancy in the data. Redundancy in this case is a characteristic which is related to predictability, randomness, smoothness, and so on. In the second category, compression is achieved by an energy preserving transformation of the given image into another array such that maximum information is packed into a minimum number of samples.

The source of a digital image may be a 3-D scene in the real world, or it may be a 2-D image generated previously, for example a photograph. In either case, the information in the source is analog in that it is continuous in both space and amplitude. To generate a digital image from a continuos one, the source is first sampled at discrete locations using...
some type of sensor (or sensors, for color images and multispectral data). These samples are termed picture elements or pixels and they are continuous over some finite intervals. These are different strategies for the sampling locations, but the most common one is an equispaced rectangular grid. Ideally, each sample corresponds to an infinitesimally small region of the source, but because of the physical nature of sensors and associated optics, it is actually an integrated value over some finite area. The higher the number of the samples (or smaller the region corresponding to each sample), the greater the resolution of the sampled image. The number of sampled locations per unit area defines the sampling rate of the system, and this sampling rate should satisfy the Nyquist sampling theorem; for example the sampling rate should be at least twice the highest spatial frequency component of the source [29].

Digital images, in their canonical representation, generally contain a significant amount of redundancy. Image compression which is the technique of efficient coding of picture data, aims at taking advantage of this redundancy to reduce the number of bits required to represent them. In general there are three types of redundancy in digital images [29]:

1. Spatial redundancy which is due to correlation between different neighboring pixel values in the image. In other words, a great deal of information about a pixel value can be obtained by examining its neighboring pixel values.
2. Spectral redundancy, which is due to the correlation between different color planes (for example in RGB color images) or spectral bands (for example aerial photographs in remote sensing applications where there is a significant band-to-band correlation).
3. Temporal redundancy which is due to correlation between different frames in a sequence of images (it can be exploited by motion compensation coding).
It is very important to develop a variety of image compression algorithms suited to different applications. There are many approaches, but all can be categorized into two fundamental groups: lossless and lossy compression [6].

In lossless compression, the reconstructed image is numerically identical to the original image on a pixel-by-pixel basis. Obviously, lossless compression is ideally desired since no information is compromised. However, only a modest amount of compression can be achieved. The lossless image compression techniques are suitable for medical applications in which fine image details are required for diagnosis.

In lossy compression, the reconstructed image contains degradation relative to the original image. As a result, much higher compression can be achieved as compared to lossless compression. In general, more compression can be obtained at the expense of more distortion. It is important to note that these degradations may or may not virtually apparent. In fact, the term visually lossless has been used to characterize lossy compression schemes that result in no visible loss under normal viewing conditions.

The recent interest in artificial neural networks has motivated a large number of applications covering a wide range of research fields. Digital image compression is one of those areas. The efforts here have mostly concentrated in artificial neural network implementation of vector quantization and autoassociative memory representation [10],[15]. The first method is based on Kohonen’s unsupervised training algorithm called self-organization feature maps (SOFM) for vector quantization. It consists of two layers of input and output nodes interconnected by weighted links without any form of feedback. Unlike a multi-layer feed-forward network, each output node has a linear response to the sum of activations from the input node. In this algorithm, continuos-valued input vectors (blocks of image arranged as a vectors) are presented to the input nodes sequentially in time.
without specifying the desired output response. For each input vector presented, the Euclidean distance to all the output nodes are computed. The weights of the node with the minimum distance and its neighboring nodes are adjusted such that the output of these nodes are slightly enhanced. The process is repeated until a certain error criterion is reached. The latter method is based on a three-layer linear perceptron, in which the second hidden layer performs the image compression and the third layer, the reconstruction. More specifically, every pixel of an \( N \times N \) image is fed into the input of a linear artificial neural network with \( h \) hidden units and the network is trained by setting the desired output equal to the input. Using some error metric such as the mean square error criterion, a gradient method such as backpropagation is used to adjust the weights to reduce the error between the actual and desired outputs. Typically the network is trained on small size image blocks (8x8, or smaller) and tested on the desired image. If \( h < n \), then a compressed version of the input image will be available at the output of the hidden nodes.

In this chapter, a novel approach for low bit rate image coding is introduced. In this new technique, image compression is achieved first by segmenting an image into regions of different sizes based on perceptual variation in each region. The resulting blocks are then coded by a three layer neural network with a different number of hidden nodes. The network parameters are quantized and arithmetic coded.

The organization of this chapter is as follows: Section 2.2 is devoted to the discussion of the JPEG baseline system. Section 2.3 presents the quadtree segmentation method, Section 2.4 addresses image modeling using neural networks. Section 2.5 presents the quantization procedure. Section 2.6 is devoted to arithmetic coding. The experimental results are provided in Section 2.7. Section 2.8 presents conclusions.
2.2. JPEG Baseline System

A committee known as JPEG (Joint Photographic Expert Group) was formed at the end of 1986 with the purpose of developing an international standard for the compression of continuos-tone, still-frame, monochrome and color images [40]. A major motivation for the formation of JPEG was the advent of multimedia services on the 64 kbits/s Integrated Services Digital Networks (ISDN) [29]. The goal of this committee was to develop a method for continuos-tone image compression which is state of art with regard to compression rate and accompanying image fidelity, over a wide range of image quality ratings. The method should also be applicable to practically any kind of continuos-tone digital source image without any restriction on images of certain dimensions, color spaces, and pixel aspect ratios. The standard should also be practical for a wide variety of applications such as desktop publishing, photo-videotex, graphic arts, photojournalism, medical systems, and many others [40]. Although no standard existed previously in these areas, JPEG members had the strong belief that the requirements of most of these applications could be satisfied by a common, general-purpose image compression technique. The proposed JPEG standard consists of three main components [28]: 1) A baseline system that provides a simple and efficient algorithm that is adequate for most image coding applications. 2) A set of extended system features that allows the baseline system to satisfy a broader range of applications. Among these optional features are 12 bits/pixel input, progressive sequential and hierarchical build-up, and arithmetic coding. In hierarchical encoding, the image is encoded at multiple resolutions, so that lower-resolution versions may be accessed without first having to decompress the image at its full resolution. 3) An independent lossless compression in which the image is encoded to guarantee exact recovery of every source image sample value (even though the result is low compression compared to the lossy modes) for applications requiring that type of compression (medical images).
Transform image coding using the discrete cosine transform (DCT) was adopted in JPEG standards. Figures 2.1 and 2.2 show the key processing steps which are the heart of the DCT-based modes of operation. These figures illustrate the special case of single-component (grayscale) image compression. One can grasp the essentials of the DCT-based compression by thinking of it as essentially compression of a stream of 8 by 8 blocks of grayscale image samples. Color image compression can then be approximately regarded as compression of multiple grayscale images, which are either compressed entirely one at a time, or are compressed by alternately interleaving 8 by 8 sample blocks from each in turn [40].

Figure 2.1. DCT Based Encoder

Figure 2.2. DCT Based Decoder
The following is a brief description of the JPEG baseline system [29].

- The original image is partitioned into 8 by 8 pixel blocks and each block is independently transformed using the DCT. The DCT step lays the foundation for achieving data compression by concentrating most of the signal in lower spatial frequencies.

- All transformed coefficients are normalized (weighted) by applying a user-defined normalization array that is fixed for all blocks. Each element of the normalization array can be any integer value from 1 to 255 and is passed to the receiver as part of the header information that is required for every image. Up to four different normalization arrays can be specified; for example, different normalization arrays may be used for the different color components of a color image. The normalized coefficients are then uniformly quantized by rounding to the nearest integer. The normalization array can be viewed as scaling the quantizer so as to control the amount of quantization error introduced in each coefficient.

- The DCT coefficient values can thus be regarded as the relative amounts of the 2D spatial frequencies contained in the 64-point input signal. The coefficient with zero frequency in both dimensions is called the DC coefficient and is treated separately from the 63 AC coefficients. After quantization, this coefficient is encoded with a lossless DPCM scheme using the quantized DC coefficient from the previous block as a 1-D predictor. For the baseline system, up to two separate Huffman tables for encoding the resulting differential signal can be specified in the header information.

- The quantization of the AC coefficients produces many zeros, especially at higher frequencies. To take advantage of these zeros, the 2-D array of the DCT coefficients is formatted into a 1-D vector using a zigzag reordering. This rearranges the coefficients in
approximately decreasing order of their average energy with the aim of creating large runs of zero values.

-To encode the AC coefficients, each nonzero coefficient is first described by a composite 8-bit value, denoted by I, of the form:

\[ I = 'SSSSNNNN' \]

The four least significant bits, 'NNNN' define a category for the coefficient amplitude. The values in category \( k \) are in the range \((2^{k-1}, 2^k - 1)\) or \((-2^k + 1, -2^{k+1})\), where \( k \) is between one and ten for the baseline system. The coefficient values contained in each category are shown in Table 2.1. Given the category, it is then necessary to send additional \( k \) bits to completely specify the sign and magnitude of a coefficient within that category. The four most significant bits in I give the position of the current coefficient relative to the previous nonzero coefficients. The runlengths specified by 'SSSS' can be ranged from 0 to 15, and a separate symbol, \( I = '11110000' \), is defined to represent a runlength of 16 zero coefficients. If the runlength exceeds 16 zero coefficients, it is coded by using multiple symbols. In addition, a special symbol, \( I = 0 \), is used to code the end of block (EOB), which signals that all the remaining coefficients in the block are zero.

-At the decoder, after the encoded bit stream is Huffman decoded and the 2-D array of quantized DCT coefficients is recovered, each coefficient is denormalized by multiplying it by the corresponding component of the normalization matrix. The resultant array is inverse DCT transformed to yield an approximation to the original image block. The resulting reconstruction error depends on the amounts of quantization, which is controlled by the normalization matrix.
2.3. Segmentation

Natural gray level images can usually be divided into different size blocks with variable amounts of detail and information. Such segmentation of the image is useful for efficient coding of image data. Quadtree decomposition can provide the flexibility of a variable block size partitioning while avoiding excessive overhead or side information [39]. It is an efficient data structure that provide an effective compromise between the accuracy with which the region boundaries are determined and the number of bits required to specify the segmentation. Other methods of image segmentation, such as region growing, more precisely isolate statistically homogeneous segments; however, in these techniques the

<table>
<thead>
<tr>
<th>Category</th>
<th>AC Coefficient Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1, 1</td>
</tr>
<tr>
<td>2</td>
<td>-3, -2, 2, 3</td>
</tr>
<tr>
<td>3</td>
<td>-7, ..., -4, 4, ..., 7</td>
</tr>
<tr>
<td>4</td>
<td>-15, ..., -8, 8, ..., 15</td>
</tr>
<tr>
<td>5</td>
<td>-31, ..., -16, 16, ..., 31</td>
</tr>
<tr>
<td>6</td>
<td>-63, ..., -32, 32, ..., 63</td>
</tr>
<tr>
<td>7</td>
<td>-127, ..., -64, 64, ..., 127</td>
</tr>
<tr>
<td>8</td>
<td>-255, ..., -128, 128, ..., 255</td>
</tr>
<tr>
<td>9</td>
<td>-511, ..., -256, 256, ..., 511</td>
</tr>
<tr>
<td>10</td>
<td>-1023, ..., -512, 512, ..., 1023</td>
</tr>
</tbody>
</table>
image has to be examined first [21]. This fact implies that a very large number of bits may be needed to represent the shape and location information. **Quadtree**, on the other hand, require only a small overhead rate by restricting the shape and the number of possible sizes of the final regions from a predetermined set of options.

**Quadtree** decomposition can be done either by top-down or bottom-up procedures [34]. In **top-down** approach, the segmentation starts with the largest allowable block size. If the block is not homogenous in the property of interest, it will be segmented to the smaller block sizes. This process continues until the smallest allowable block size is achieved. In bottom-up procedure, first the largest starting block size is segmented to the smallest allowable block sizes. Then the algorithm attempts to group the low detail regions into blocks that are as large as possible, while keeping the high detail regions into small size blocks. In this work, the top-down **quadtree** segmentation is implemented. In Figure 2.3, both **top-down** and bottom-up **quadtree** decomposition procedures are illustrated.

A **quadtree** is a tree structure in which each nonterminal node has four branches emanating from it. These branches point to nodes that are children of the parent node. In our application of image coding, every node corresponds to a **subblock** of the image. The size and location of the **subblock** is determined by its position on the tree. The four children of a particular parent node represent the four subblocks obtained by splitting the parent block into four equal-size squares. The tree itself consists of several stages, or levels of nodes, where a node at the nth stage represents a one-quarter **subblock** of the parent image block from the previous stage.
Figure 2.3. The structure of quadtree segmentations. a) original image, b) top-down quadtree, c) bottom-up quadtree.

The procedure for quadtree segmentation of an image starts at an initial node, the root, corresponding to some particular image block with a relatively large size, and proceed through the tree from level to level. An analysis of test images showed that the blocks larger than $32 \times 32$ are almost always inhomogeneous, and our algorithm thus starts with a partition of the image into block of this size. At each node, a test is performed to determine whether the block represented by that node is homogeneous in the property of interest. If the test is positive, then the node becomes a leaf, or terminal node.
Otherwise, the segmentation proceeds to the next level and the four children of the node are each examined in turn. The procedure always terminates when it reaches the node level corresponding to a specific minimum allowed block size. Figure 2.4 shows the tree structure of quadtree segmentations.

Our segmentation algorithm divides the image into variable-size blocks based on variance. The following procedure is used to determine the quadtree segmentation thresholds. Starting with the largest allowable blocks, first the histogram of variance is analyzed. Then, a variance threshold is chosen such that the blocks with variance greater than the threshold are segmented. To avoid blockiness, the value of this threshold is smaller for the large block sizes. This process continues until the smallest allowable block size is achieved. Since there is a direct relationship between the bit rate and the block size, the selection of thresholds is an important part of the segmentation process. The number of parameters for each block which are sent to the decoder is fixed and is independent of block size. So as the number of blocks (resulted from segmentation) grows, the bit rate also increases.

![Figure 2.4. The tree structure of quadtree segmentations of the final block.](image)
Figure 2.5. Quadtree segmentation of the test image Lenna.
On the other hand, the larger the number of blocks (resulting from segmentation), the higher is the signal-to-noise ratio. So overall it can be concluded that the final segmentation map resulted from quadtree segmentation is dictated by the combination of desired bit rate and the desired signal-to-noise ratio. Figure 2.5 demonstrates a typical segmentation map for the 512x512 test image Lenna.

2.4. Image Modeling By A Neural Network

Once the image is segmented, every block \( B \) is modeled in the following procedure. Let \( f(z) \) denote the image intensity at the pixel location \( z=(x,y) \) and let \( \hat{f}(z) \) denote the coded value. \( \hat{f}(z) \) can be expressed as:

\[
\hat{f}(z; \Theta) = \sum_{i=1}^{k} \gamma_i \sigma(\alpha_i \cdot z + \beta_i)
\]

Where \( \Theta = (\alpha_i; \beta_i; \gamma_i) \) is the set of parameters of the model, \( \sigma(\bullet) \) denotes the sigmoid function, and \( A \cdot B \) represents dot or scalar product of vectors \( A \) and \( B \). Optimal parameters of the model are obtained as:

\[
\Theta^* = \arg\min_{\Theta} \sum_{z \in B} d(f(z), \hat{f}(z; \Theta))
\]

Where \( d(\bullet, \bullet) \) is some desired error metric.

The neural network implementation of the above model is shown in Figure 2.6. This is a typical 3 layer (input layer, hidden layer, output layer) network which has a hidden layer sigmoidal neurons. The hidden layer receives inputs directly and then broadcast their outputs to a layer of linear neurons which compute the network output. The input to the network is the pixel location and corresponding desired output is the image intensity at that location.
The number of samples for training of the network depends on the block size. For the image block size of 32x32, 16x16 and 8x8, the corresponding number of inputs are 1024, 256, 64, respectively. It should be noted that the size of the input and output vectors are 2 x 1 and 1 x 1, respectively, and are independent of block sizes.

Parameter $k$ is the size of the hidden layer for the desired neural network. It reflects the complexity of the fitted model and is selected based on some measure of "activity" in the block. In this work, variance of each block is used to measure the amount of activity in a block. A network with smaller hidden layer size (small $k$) is required for the blocks which are extracted from a smooth part of the image. On the other hand, a large $k$ is required for the blocks which are heavily detailed. So parameter $k$ determines the number parameters which have to be sent to the decoder. In other words, this parameter controls the compression ratio and the quality of the reconstructed image.
We implemented backpropagation learning rule to train our network. The backpropagation learning adjusts the weights and biases of the networks in order to minimize the sum squared error between the actual outputs and the desired outputs. This is done by continually changing the values of the network weights and biases in the direction of steepest descent with respect to error.

To train a network, input vectors are presented to the network. The actual output of the network is calculated. The sum of the squared errors is then found. If the sum squared error for all training vectors is less than the error goal, training stops. Otherwise, the derivatives of error (called delta vectors) are calculated for the network's output layer and then backpropagated through the network until delta vectors are available for each hidden layer. This backpropagation of delta vectors is where the backpropagation learning technique gets its name [10]. Finally the weights and biases of each layer are updated using the layer's delta vectors. The backpropagation algorithm is discussed in more detail in Chapter 1.

2.5. Quantization

The optimum parameters (weights and biases) for each block must be quantized before encoding. This means that the range of values of the parameters must be divided into intervals and all the values within an interval must be represented by a single level [27].

Let the output levels of the quantizer be denoted by \( q_1, q_2, \ldots, q_k \) and the input intervals be denoted by the decision levels \( z_1, z_2, \ldots, z_{k+1} \). This indicates that any weights or biases with a value anywhere between \( z_k \) and \( z_{k+1} \) is assigned the value \( q_k \) at the output of the quantizer. The mean square error between the input and output of Lloyd-Max quantizer can be written as [31]
where $p(z)$ is the probability density function for the weights or biases which is represented by the continuous variable $z$. For a given $k$ (number of output levels), we would like to determine the output levels ($q_k$'s) and the decision levels ($z_k$'s) such that the mean square quantization error is minimum. This involves setting the partial derivatives of error with respect to the $z_k$ and $q_k$. This minimization of the error gives the transition levels that lie half way between the reconstruction levels and the reconstruction levels that lie at the center of mass of the density in the transition intervals. Mathematically, they are given by the solution of the simultaneous nonlinear equations as

$$
  z_k = \frac{q_{k-1} + q_k}{2}, \quad k = 2, 3, ..., k
$$

$$
  q_k = \frac{\int_{z_k}^{z_{k+1}} z p(z) \, dz}{\int_{z_k}^{z_{k+1}} p(z) \, dz}
$$

Based on the distribution of the weights and biases and their dynamic ranges, separate Lloyd-Max quantizers were designed for each block. In this work, Gaussian Lloyd-Max quantizers with 5 or 6 bits provided a good fit to the data.

2.6. ARITHMETIC CODING

The resulting quantized parameters can be coded in a number of ways. The effectiveness of any coding scheme can be measured in terms of the entropy of the message, usually expressed in bits/symbol. Shannon's fundamental theorem of coding
states that, given messages randomly generated from a model, it is impossible to encode them into less bits (on average) than the entropy of that model [41]. Various coding schemes such as Huffman coding and arithmetic coding were considered [19], [17]. Huffman coding can encode source symbols with an average bit rate arbitrarily close to the source entropy, but it suffers from several fundamental limitations. For example, consider a source that has one symbol whose probability of occurrence is near one, and hence the other symbols have low probabilities. Since the shortest possible code word length is one, the bit rate for this source will be approximately 1 bit/symbol even though its entropy is significantly less than this.

In this work, arithmetic coding was chosen based on the following [6]:

1. Arithmetic coding can approach the entropy limit in coding efficiency.
2. Arithmetic coding requires only one pass through the data.
3. The source modeling and information encoding are separated.
4. Arithmetic coding is generally faster than Huffman coding.
5. An Arithmetic coder requires no a priori analysis of the data set to determine bit allocation.

The arithmetic coder assigns one "code word" to each possible input data set. The code words consist of half-open subintervals of the half-open unit intervals [0,1), and are expressed by specifying enough bits to distinguish the subinterval corresponding to the actual data set from all other possible subintervals. The idea is to assign shorter code words to larger subintervals and thus more probable input data sets. In practice, the subinterval is updated incrementally using the probabilities of the individual events, with bits being output as soon as they are known.
In practice, the probability of each possible event has to be estimated at each point in the coding process. A statistical coder must work in conjunction with a modeler to accomplish this task. The probability model does not have to describe the details involved in generating the data. It merely has to provide a probability distribution for the data items. The more accurate the probabilities are, the better the compression will be. On the other hand, if the probabilities are wildly inaccurate, the file may even be expanded rather than compressed. To obtain maximum compression of a file, we need both a good probability model, and an efficient way of representing the probability model.

To ensure the decodability, the encoder is limited to the use of model information that is available to the decoder. The models can be adaptive, semi-adaptive and nonadaptive. In the case of adaptive models, the probability of each event is estimated dynamically based on all events that proceed it [17]. The semi-adaptive case requires a preliminary pass of the input data to gather statistics. In the case of nonadaptive models, a fixed probability is used for all the files. The performance of nonadaptive models can be poor. Adaptive codes allow one-pass coding but require a more complicated data structure. Semi-adaptive codes require two passes and transmission of model data as side information. If the model data are transmitted efficiently, semiadaptive codes can provide slightly better compression than adaptive codes, but in general the cost of transmitting the model is about the same as the learning cost in the adaptive case.

The basic algorithm for arithmetic coding can be summarized as follows [17]:

1. We begin with a current interval initialized to half-open interval $0 \leq x < 1$.
2. For each event in the file, the current interval is subdivided into subintervals, one for each possible event. The size of an event's subinterval is proportional to the estimated
probability of that event. The subinterval corresponding to the next event is selected to *become* the new current interval.

(3) *At* the last step, enough bits will be used to distinguish the final current interval from all other possible final intervals.

The length of the final subinterval is clearly equal to the product of the probabilities of the individual events. However, the decoder will face the problem of detecting the end of file, to determine when to stop decoding. To solve this problem, either a *special* end-of-file event is coded at the end of the file, or some external indications of the file's length is used. Either method adds only a small amount to the code length.

To illustrate different steps in the development of arithmetic coding, an example is presented [41]. Suppose we have a file consisting of six events which are denoted by \{a, e, i, o, u, !\}. The fixed probability model which is used for this *example* is shown in table 2.2. Let us assume the actual message to be encoded is the sequence *eaii!*. Initially, both the encoder and the decoder know the range for the message is the entire interval \([0, 1)\). Every time the encoder sees a new symbol, the interval is refined. The first symbol which encoder sees is *e*. The range which the model allocates for this symbol is \([.2, .5)\).

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>PROBABILITY</th>
<th>RANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0.2</td>
<td>([0, .2))</td>
</tr>
<tr>
<td>e</td>
<td>0.3</td>
<td>([.2, .5))</td>
</tr>
<tr>
<td>i</td>
<td>0.1</td>
<td>([.5, .6))</td>
</tr>
<tr>
<td>0</td>
<td>0.2</td>
<td>([1.6, .8))</td>
</tr>
<tr>
<td>u</td>
<td>0.1</td>
<td>([1.8, .9))</td>
</tr>
<tr>
<td>!</td>
<td>0.1</td>
<td>([1.9, 1.0))</td>
</tr>
</tbody>
</table>
So the encoder narrows the range to the corresponding range of the symbol e which is [.2,.5). The next symbol, a, will narrow this range to the first one-fifth of it, since a has been allocated [0,.2). This symbol refines the range to [.2,.26) which has the length of .06. Since the previous range was 0.3 units long, one-fifth of that is .06. The next symbol which the encoder sees is i. This symbol has probability of 0.1 and range of [.5,.6). When this image is applied to the previous range, a smaller range of [.23,.236) is obtained. Proceeding in this way, the encoded message builds up as follows:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>initially</td>
<td>[0, 1)</td>
<td></td>
</tr>
<tr>
<td>after seeing e</td>
<td>[.2, .5)</td>
<td></td>
</tr>
<tr>
<td>after seeing a</td>
<td>[.2, .26)</td>
<td></td>
</tr>
<tr>
<td>after seeing i</td>
<td>[.23, .236)</td>
<td></td>
</tr>
<tr>
<td>after seeing i</td>
<td>[.233, .2336)</td>
<td></td>
</tr>
<tr>
<td>after seeing !</td>
<td>[.23354, .2336)</td>
<td></td>
</tr>
</tbody>
</table>

Another representation of arithmetic coding process is shown in Figure 2.7. In this configuration, the ranges expanded to full height at every stage and marked with a scale that gives the end points as numbers.

The decoding process starts with analyzing the last subinterval. Suppose the decoder knows the final range, [.23354, .2336) (even a single number within the range suffice). Since the final range lies entirely within the space the model of table I allocates fore, it can immediately deduce that the first character was e. So again starting with initial interval of [0,1), the refined interval after detecting e is [.2, .5). The new subinterval makes it clear that the second character is a. This will reduce the interval to [.2, .26). Proceeding in this way, the decoder can identify the coded sequence. In the above example, the symbol !(End-of-File symbol) is used to terminate the message. When the decoder sees this symbol, it stops decoding.
Figure 2.7. Representation of the Arithmetic Coding Process
2.7. Experimental Results

Simulations have been carried out to demonstrate the effectiveness of the proposed algorithm. The test image Lenna is shown in Figure 2.8. An analysis of the test images showed that the blocks larger than 32x32 are almost inhomogeneous. Thus, our quadtree segmentation algorithm starts with a partitioning of the image into blocks of 32x32. These blocks are further subdivided into smaller subblocks based on a "measure of activity" within each block. Because of ease of implementation, variance is chosen as the measure of activity within each block. To find the quadtree segmentation threshold, first the entire image is segmented into blocks of 32x32. Then, the histogram of variance is obtained. By analyzing the histogram, a threshold is selected. For the test image Lenna, the selected threshold for 32x32 blocks was 100. Blocks with variance higher than the threshold are segmented into 16x16 blocks and the same procedure is repeated. It is desirable to keep the number of 32x32 blocks as few as possible to avoid blockiness. Since the compression ratio for blocks smaller than 8x8 is not significant, our segmentation algorithm stops splitting the blocks after this point. To come up with different segmentation maps, several simulations with different segmentation thresholds have been performed. At each simulation, the corresponding bit rate has been calculated. Considering the tradeoff between the bit rate and the number of total blocks, a set of segmentation thresholds were selected. The values of these thresholds for the block sizes of 32x32, and 16x16 are given by 100, 1500, respectively. The segmentation map based on the image variance measure for our test image is shown in Figure 2.5. The number of segmented blocks resulting from the variance based quadtree segmentation algorithm for the test image Lenna is shown in Table 2.3.
Table 2.3. The number of segmented blocks produced by quadtree segmentation.

<table>
<thead>
<tr>
<th>BLOCK SIZE</th>
<th>8x8</th>
</tr>
</thead>
<tbody>
<tr>
<td>32x32</td>
<td>53</td>
</tr>
<tr>
<td>16x16</td>
<td>667</td>
</tr>
</tbody>
</table>

Once the image is segmented, the backpropagation learning algorithm is invoked to construct a distinct code for each block. In this approach, a three layer neural network is designed for each block by training with the backpropagation algorithm. The inputs to the network are the pixel locations. The desired output for each input location is the corresponding pixel intensity. The size of the input vector and output vector is 2x1 and 1x1, respectively and is independent of the block size. The inputs are normalized to values between 0 and 1. The initial weights and biases of the network are initialized to small values between +1 and -1. The number of training iterations is set to 500 epochs and is chosen independent of the block size.

The most important parameter in the design of the neural network is the size of the hidden layer which controls the compression ratio and the quality of the reconstructed image. The larger is the size of the hidden layer nodes, the higher is the bit rate and therefore the better is the signal to noise ratio of the encoded block. In turn, it should be evident that the blocks extracted from smooth parts of the image are "simpler" to code with respect to blocks, which are heavily detailed. The simpler blocks can then be elaborated by a neural network with a small number of hidden nodes, whereas higher number of hidden nodes are required by more complicated blocks.
We used the following procedure to determine the number of hidden nodes for each layer. First, the histogram of variances for all same size blocks is analyzed. Then a set of thresholds is chosen to classify the blocks into different categories based on the amount of activity in the blocks. Depending on the block size and the category, a neural network with three to eight hidden nodes is designed to train each block. Since compression for the blocks of 32x32 is more craving, the network for these blocks have the highest number of hidden nodes. Overall the number of hidden nodes controls the number of the parameters which are sent to the decoder. The number of fixed size blocks as a function of number of hidden layer nodes is shown in Table 2.4. The blocks of 32x32 pixels are fitted the largest number of hidden layer nodes (six or eight).

The optimum weights and biases were quantized using Lloyd-Max quantizers. In this experiment, Gaussian Lloyd-Max quantizers are used to quantize the optimum network parameters. The quantized parameters were then arithmetic encoded. At the decoder, the inputs (pixel locations) and the decoded network parameters are used to reconstruct all pixel values. The decoding procedure of this algorithm is very fast and efficient.

Table 2.4. The number of fixed size blocks as a function of number of hidden layer nodes.

<table>
<thead>
<tr>
<th>BLOCK SIZES</th>
<th>NUMBER OF HIDDEN LAYER NODES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td>32x32</td>
<td>0</td>
</tr>
<tr>
<td>16x16</td>
<td>137</td>
</tr>
<tr>
<td>8x8</td>
<td>95</td>
</tr>
</tbody>
</table>
In order to compare the coding fidelity among the different schemes, it is necessary to use an objective measure of reconstruction quality. Thus, following common practice, we use the peak signal to noise ratio defined as follows: Let \( f(i, j) \) be a pixel at the \( i, j \) coordinates of the source image and let \( \hat{f}(i, j) \) be a pixel at the \( i, j \) coordinates of the reconstructed image. The peak signal to noise ratio for an eight bit (0-255) image is defined by

\[
\text{PSNR} = 10 \log \left( \frac{255^2}{\sum_{i=1}^{N} \sum_{j=1}^{N} (f(i, j) - \hat{f}(i, j))^2} \right)
\]

where: \( N \times N \) is the size of the image \([34],[39]\). It is important to note that PSNR does not necessarily imply a higher subjective reconstructed image quality.

Simulations have been carried out to demonstrate the effectiveness of the proposed algorithm. The coded test image at a bit rate of .25 bit per pixel (bpp) with PSNR of 29.65 dB is shown in Figure 2.9. For comparison, the JPEG algorithm with DCT coding was used to encode the test image at the bit rate of .25 bpp. The PSNR of the corresponding PEG decoded image was found to be 30 dB. The PEG decoded image with bit rate of .25 bpp is shown in Figure 2.10. The proposed algorithm provides images which are compatible in bit rate and PSNR. The subjective quality of images are also very close.

2.8. Conclusions

A novel approach for low bit rate model-based image compression has been presented in this chapter. It has been demonstrated that quadtree based image segmentation can be an effective and efficient mechanism for identifying blocks of distinct perceptual significance and thereby allowing different coding strategies that are perceptually suited to the individual
segment categories. We introduced a new variable-rate coding technique which is based on the amount of activity in each individual block. Central to our coding approach is the use of the variable-rate coding that conceptually corresponds to the idea of designing neural networks with different hidden layers. In this approach, the "simpler" blocks are required a neural network with a small number of hidden layer nodes. On the other hand, the more detailed blocks require a network with a large number of hidden layer nodes. Perceptually, we have obtained high quality image reproduction at rates of .25 bpp. Objective measurements of performance using PSNR, while of moderately limited use in assessing quality, has nevertheless given values comparable to the JPEG decoded image.

In summary, this work has introduced a new direction of research on model based image compression using neural networks. It also demonstrates the potential of segmentation based image coding in which the segmentation step is accomplished by the quadtree method. Substantially improved performance with further development of this approach is reported in the next chapter.
Figure 2.8. Test image Lenna.
Figure 2.9. The reconstructed test image Lenna with the proposed algorithm with a bit-rate of .25 bpp and PSNR of 29.65 dB.
Figure 2.10. The JPEG decoded image with bit rate of .25 bpp and PSNR of 30 dB.
CHAPTER 3. IMAGE CODING WITH NEURAL NETWORKS AND PROJECTION PURSUITS

3.1. Introduction

A new algorithm for image compression using a neural network was presented in chapter 2. The new algorithm produces a compressed image which is compatible with the corresponding JPEG reconstructed image. In this approach, first the image is segmented to variable block sizes. Depending on the block size and the amount of activity within the block, a neural network with a fixed number of hidden nodes is chosen. The simpler blocks require a neural network with a small number of hidden layer nodes whereas the more detailed blocks require a network with a large number of hidden layer nodes. This algorithm can be further improved if the optimum number of hidden nodes for each block is known.

In this chapter, another novel approach for low bit rate image coding is presented. In this new technique, image compression is achieved first by segmenting an image into regions of different sizes based on one of two possible measure of activity (perceptual variation and PSNR of the prereconstructed block) and then constructing a distinct code for each block by invoking the theory of neural networks and projection pursuits. The implementation of this algorithm allows one to adaptively find the optimum number of parameters which is required to code a block. The process is terminated when the desired PSNR or bit-rate is achieved.
In order to compare the coding fidelity among the different schemes, it is necessary to use an objective measure of quality. Thus, following common practice, we use the PSNR defined in Chapter 2. It is important to note that PSNR does not necessarily imply a higher subjective quality.

The organization of this chapter is as follows: Section 3.2 is devoted to image segmentation. Section 3.3 presents the theory of neural networks and projection pursuits. Section 3.4 addresses the steepest descent algorithm. Section 3.5 presents the reduced gradient method. The experimental results are provided in Section 3.6. Section 3.7 presents the conclusions.

3.2. Quadtree Segmentation

Efficient coding of image data requires segmentation which divides the image into different size blocks with variable amounts of detail and information. Quadtree segmentation is a compromise between the fixed size, fixed shape and arbitrarily shaped region decomposition. It is done either by a top-down or a bottom-up procedure. Briefly, the top-down approach of quadtree decomposition can be stated as follows [15], [34]. The procedure starts from a pre-segmented image with blocks of NxN pixels, the largest block allowed in the segmentation process. At each block, a test is performed to determine whether the block is homogeneous in the property of interest. If the test is negative, each of these NxN blocks are further subdivided into four smaller blocks of \( \frac{N}{2} \times \frac{N}{2} \) pixels. This subdivision continues until a maximum predefined decomposition stage \( k \), corresponding to a blocksize of \( \frac{N}{2^k} \times \frac{N}{2^k} \) pixels, is reached.

Quadtree decomposition is attractive for the following reasons [39]:

\[
\text{Quadtree decomposition is attractive for the following reasons [39]:}
\]
1) it is relatively simple to implement as compared to other methods of image representation.

2) it adaptively decomposes the image; i.e., it divides the image into regions with sizes depending on the activities in the blocks. This leads to variable-rate image coders that change the coding resolution (in bits used per unit area) according to the local character and importance of the image being coded.

The process of selecting quadtree segmentation thresholds is an important part of the segmentation process. The total number of blocks resulting from segmentation is controlled by these thresholds. When the threshold values are high, the number of blocks which are segmented to smaller blocks are less. Therefore, the algorithm is unable to accurately isolate the homogenous segments. In this chapter, two measures of activity are used to distinguish the smooth blocks from the high detailed ones. The first method is based on the variance of each block. The process of selecting the variance based quadtree segmentation thresholds is the same as Chapter 2. Figure 2.5 demonstrates a typical variance based quadtree segmentation map for 512x512 test image Lenna. Figure 3.1 demonstrates the same quadtree segmentation map for 512x512 test image Pepper.

The second proposed method for quadtree segmentation is based on the PSNR of the reconstructed block. In this case, the segmentation process starts by selecting three PSNR thresholds. The first threshold is a global desired PSNR for the reconstructed image. The other two are the segmenting thresholds for the blocks of 32x32 and 16x16. The value of the threshold for 32x32 blocks is close to the global PSNR threshold. After selecting the thresholds, starting with the largest allowable block size, a neural network with only one hidden layer node is fitted to the block. The obtained network parameters are used to reconstruct the block. The PSNR of the original block and the reconstructed block is
Figure 3.1. The variance based segmentation of the test image Pepper.
Figure 3.2. The PSNR based **quadtree** segmentation of the test image: **Lenna**.
Figure 3.3. The PSNR based quadtree segmentation of the image Pepper.
computed. If the PSNR of the reconstructed block is less than the PSNR ratio threshold, the block is segmented to four smaller blocks. The process continues until the smallest allowable block size is achieved.

The selection of segmentation threshold for the case of PSNR based quadtree segmentation is an important part of the segmentation process. They control the bit-rate and the quality of the reconstructed block. If the selected threshold values are close to the global desired PSNR threshold, the number of segmented blocks will be high and hence the bit-rate and the PSNR increases. So overall it can be concluded that the final segmentation map is dictated by combination of desired bit-rate and desired PSNR. Figure 3.2 demonstrates a typical PSNR-based quadtree segmentation map for the 512x512 test image Lena. Figure 3.3 demonstrates the same quadtree segmentation map for the 512x512 test image Pepper.

3.3. Image Coding With Neural Networks And Projection Pursuits

Once the image has been segmented into various size regions, each block is coded by using a neural network together with projection pursuits. Projection pursuits is an efficient iterative function approximation method. In the first step of the iteration, the current desired image is the original image, and in step k, the current desired image is the residual or the error image that results from subtraction of the linear combination of all the (k-1) previous approximations from the original image. Various measures of error such as mean square error and absolute error can be used to assess the quality of the approximation. Due to mathematical tractability, mean square error is chosen here. The popularity of squared error is partly due to the wealth of theory and numerical methods available for the analysis of systems which are optimal in the sense of minimizing mean squared error. The technique
of neural networks with projection pursuits is similar to the method of parallel, self-
organizing neural networks with continuous inputs and outputs [9], [12].

For the proposed procedure, the following quantities are defined:

\[ f(x, y) = \text{Image intensity at the coordinates } x \text{ and } y. \]
\[ \hat{f}(x, y) = \text{Estimated image intensity at the coordinates } x \text{ and } y. \]
\[ r_k(x, y) = \text{The residual image at iteration } k, \text{ and at the coordinates } x \text{ and } y. \]
\[ \Theta_k = \{ a_k, \beta_k, \gamma_k \} = \text{The set of parameters at iteration } k. \]
\[ d(\ldots) = \text{The desired error metric which is typically the mean square error criterion.} \]
\[ g(.) = \text{The sigmoidal transfer function.} \]
\[ BR = \text{Bit rate.} \]

The scalar product of vectors \( A \) and \( B \) is denoted by \( A \cdot B \), \( A^T \) represents the transpose of vector \( A \), and \( z \) represents \( z^T = [x \ y] \).

The algorithm consists of the following:

for every block \( B \) in the segmentation map, do

\[
\text{while } (\text{PSNR}_k \leq \text{PSNR}_{\text{desired}} \text{ or } BR_k \leq BR_{\text{desired}}) \{ \]

Let \( r_0 = f \);

at iteration \( k \), select the optimal parameter vector \( \Theta_k \):

\[
\Theta_k = \arg \min_{a_k, \beta_k, \gamma_k} \sum_{(x,y) \in B} d(r_{k-1}(x,y), \gamma_k g(\alpha_k^T z + \beta_k))
\]

\[
\hat{f}_k = \sum_{i=1}^k \gamma_i g(\alpha_i^T z + \beta_i);
\]

\[
r_k = f - \hat{f}_k;
\]

compute the \( \text{PSNR}_k \) and \( BR_k \), at iteration \( k \} \]

\[
\hat{f}_k = \sum_{i=1}^k \gamma_i g(\alpha_i^T z + \beta_i);
\]

end.
The neural network implementation of the above model is shown in Figure 3.5. This is a typical three layer (input layer, hidden layer, output layer) network which has a hidden layer of sigmoid neurons. The hidden layer receives inputs directly and then broadcasts its outputs to a layer of linear neurons which compute the network output. The input to the network is the pixel location and the corresponding desired output is the image intensity at that location.

To find the optimum parameters for each block, two different approaches are implemented. The first approach is based on the backpropagation algorithm (steepest descent) [16]. The second approach is based on the Gams-Minos algorithm (reduced gradient) [7]. With both approaches, a three layer neural network is designed for each block. Then, the backpropagation learning rule is used to train the network for every block. The training procedure for each block starts with a neural network which has only one hidden layer node. After training, the optimum weights and biases are obtained. Using the input pixel values and optimum weights and biases, the block is reconstructed. The PSNR ratio between the reconstructed block and the original block is computed. If this value is more than some predefined global desired PSNR ratio threshold, the training stops. Otherwise, the difference between the original pixel values and reconstructed ones is computed. This error image is treated as an input for another network which has only one hidden layer. After training, the same procedure is repeated. The approximation process is terminated when the overall error drops below the desired threshold or the desired bit-rate is achieved.

The second approach for finding the optimum parameters of each network is based on the Gams-Minos optimization algorithm [7]. The algorithm finds the optimum parameters of
the model which best fits to the pixel values in a block. The steepest descent and reduced gradient algorithms are further discussed in the following two sections.

**Projection pursuits** is an efficient algorithm for image coding which works by approximating the image progressively. There are two thresholds for controlling the quality of the reconstructed image. The bit rate is controlled by the number of stages generated. The quality of the reconstructed image (in the mean square sense) is controlled by the desired PSNR. In chapter two, the proposed image coding algorithm always trained with a network which has a predefined number of hidden layers. In the projection pursuit algorithm, it is not required to define the number of hidden layers for the network. In other words, the network progressively find the best size such that the reconstructed image satisfies one of the predefined PSNR or bit rate thresholds. The blocks extracted from a smooth part of an image require a small number of iterations compared to the blocks which are heavily detailed.
3.4. Steepest Descent Algorithm

One of the oldest and most widely known methods for minimizing a function of several variables is the method of steepest descent (often referred to as the gradient method) [25]. The method is extremely important from a theoretical viewpoint, since it is one of the simplest for which a satisfactory analysis exists. More advanced algorithms are often motivated by an attempt to modify the basic steepest descent technique in such a way that the new algorithm will have superior convergence properties. The steepest descent algorithm is also used in the original backpropagation algorithm.

The method of steepest descent is defined by the following iterative equation:

\[ x_{k+1} = x_k - \alpha_k \, g_k \]

where \( \alpha_k \) is a nonnegative scalar minimizing \( f(x, -\alpha_k g_k) \), \( g(x_k) = \nabla f(x_k)^T \), and \( \nabla f(x) \) denotes the gradient of \( f(x) \). In words, from the current point, we search along the direction of the negative gradient \( -g_k \) towards a minimum. This minimum point is taken to be \( x_{k+1} \). An explicit form for the steepest descent algorithm can be obtained when it is applied to quadratic problems. Consider

\[ f(x) = \frac{1}{2} x^T Q x - x^T b \]

where \( Q \) is a positive definite \( N \times N \) matrix. The gradient of \( f(x) \) is given by

\( \nabla f(x) = Qx - b \). The unique minimum point of function \( f(x) \) (since \( Q \) is positive definite, \( f(x) \) is strictly convex) can be found directly by setting the gradient equal to zero. This
results in a vector \( x^* \) which satisfies \( Qx^* = b \). Thus, the method of steepest descent for the quadratic case can be expressed as

\[
x_{k+1} = x_k - \alpha_k g_k
\]

where \( g_k = Qx_k - b \). For the special case of the quadratic problems, the value of \( a \) can be determined explicitly. We have by definition

\[
f(x_k - \alpha_k g_k) = \frac{1}{2} (x_k - \alpha_k g_k)^T Q (x_k - \alpha_k g_k) - (x_k - \alpha_k g_k)^T b
\]

\( \alpha_k \) can be found by differentiating the above expression as

\[
\alpha_k = \frac{g_k^T g_k}{g_k^T Q g_k}
\]

Hence, the method of steepest descent takes the explicit form

\[
x_{k+1} = x_k - \left( \frac{g_k^T g_k}{g_k^T Q g_k} \right) g_k
\]

where \( g_k = Qx_k - b \)

### 3.5. The Reduced Gradient Method

The second approach for finding the optimum parameters is based on the Gams-Minos optimization algorithm. The algorithm finds the optimum parameters of the model by using the reduced gradient algorithm [35], [18], [25]. This method is closely related to the simplex method of linear programming in that the problem variables are partitioned into basic and nonbasic groups. However, whereas the strategy of the convex simplex method
is to **move** toward a better solution point by changing the value of a single nonbasic variable, holding the others fixed, the reduced gradient strategy is to allow all nonbasic variables whose values can change favorably to do so simultaneously.

Consider the following problem:

\[
\begin{align*}
\text{minimize:} & \quad f(\bar{x}) \quad \bar{x} \in \mathbb{R}^n \\
\text{subjectto:} & \quad h_i(\bar{x}) = 0 \quad i = 1, 2, \ldots, n \\
& \quad L_j \leq x_j \leq V_j \quad j = 1, 2, \ldots, n
\end{align*}
\]

**Inequality** constraints are taken care of by subtracting nonnegative slack variables from the inequality constraints to turn them into equality constraints. Two sets of variables are distinguished in the generalized reduced gradient algorithm. If a nondegeneracy assumption holds, the m basic variables (e.g., dependent) variables comprising the set I, \( \bar{x}_i \), and the (n-m) nonbasic (e.g., independent) variables comprising the set K, \( \bar{x}_k \). The dependent variables are implicitly determined by the independent variables; hence, the objective function is a function only of the (n-m) independent variables. To illustrate the basic idea, consider an optimization problem for an objective function of just two variables subject to one equality constraint

\[
\begin{align*}
\text{Minimize:} & \quad f(x_1, x_2) \\
\text{Subject to:} & \quad h(x_1, x_2) = 0
\end{align*}
\]

For differential displacements in \( x_1 \) and \( x_2 \),

\[
df(x) = \frac{\partial f(x)}{\partial x_1} dx_1 + \frac{\partial f(x)}{\partial x_2} dx_2
\]
Furthermore,

\[ dh(x) = \frac{\delta h(x)}{\delta x_1} dx_1 + \frac{\delta h(x)}{\delta x_2} dx_2 \]

These equations are linear in the differential displacement, so that the selected differential dependent variable can be eliminated from the differential objective function.

Solve \( dh(\bar{x}) = 0 \) for \( dx_2 \).

\[ dx_2 = -\frac{\delta h(\bar{x})/\delta x_1}{\delta h(\bar{x})/\delta x_2} dx_1 \]

and substitute \( dx_2 \) into the differential objective function

\[ df(\bar{x}) = \left( \frac{\delta f(\bar{x})}{\delta x_1} - \frac{\delta f(\bar{x})}{\delta x_2} \frac{\delta h(\bar{x})}{\delta x_1} \right) dx_1 \]

to yield the reduced gradient:

\[ \frac{df(\bar{x})}{dx_1} = \frac{df(\bar{x})}{dx_1} - \frac{df(\bar{x})}{dx_2} \left[ \frac{dh(\bar{x})}{dx_2} \right]^{-1} \frac{dh(x)}{dx_1} \]

One necessary condition for \( f(\bar{x}) \) to be a minimum is that \( df(\bar{x}) = 0 \) or by analogy to the condition for an unconstrained minimum, that

\[ \frac{df(\bar{x})}{dx_1} = 0 \]

Example [18]: Minimize \( f(x) = x_1^3 + x_2^3 \)
Subject to \( h(x) = 2x_1 + x_2 - 1 = 0 \)

Let \( x, \) be the independent (nonbasic) variable and let \( x, \) be the dependent (basic) variable. The idea of the reduced gradient method is to consider, at each stage, the problem only in terms of the independent variables, since the vector of dependent variables \( x_2 \) is determined through the constraints from the vector of independent variables \( x_1. \) The objective function can be considered to be a function of \( x_1 \) only. The partial derivatives are as following:

\[
\frac{\delta f(\bar{x})}{\delta x_1} = 2x_1 \\
\frac{\delta h(\bar{x})}{\delta x_1} = 2 \\
\frac{\delta f(\bar{x})}{\delta x_2} = 2x_2 \\
\frac{\delta h(\bar{x})}{\delta x_2} = 1
\]

and the generalized reduced gradient is

\[
\frac{df(\bar{x})}{dx_1} = \frac{df(\bar{x})}{dx_1} - \frac{df(\bar{x})}{dx_2} \left[ \frac{dh(\bar{x})}{dx_2} \right]^{-1} \frac{dh(\bar{x})}{dx_1}
\]

Substituting the values for each term in the above expression yields

\[
\frac{df(\bar{x})}{dx_1} = 2x_1 - 2x_2(1)(2) = 2x_1 - 4x_2
\]
From any feasible point, a search along the constraints $h(\mathbf{x})=0$ is undertaken until $rac{df(\mathbf{x})}{dx_i} = 0$.

### 3.6. Experimental Results

Simulations have been carried out to demonstrate the effectiveness of the proposed algorithm. The two test images Lenna and Pepper are shown in Figures 3.5 and 3.6, respectively. The first step in the coding process is the **quadtree segmentation** of the image. **Quadtree** segments the image into regions of different sizes based on some measure of "activity" in each region. An analysis of test images showed that the blocks larger than 32x32 pixels are almost inhomogeneous. Thus, our **quadtree segmentation algorithm** starts with a partitioning of the image into blocks of 32x32 pixels. Two possible measures of activity are investigated in this work. The first measure is based on the **variance** of each block. The second measure is based on the **PSNR** of the reconstructed block.

The procedure for finding the variance based **quadtree segmentation** threshold is presented in chapter 2. The variance thresholds for the test image Lenna are 100 and 1500 for block sizes of 32x32 and 16x16, respectively. The variance thresholds for the test image Pepper for block sizes of 32x32 and 16x16 are 130 and 1500, respectively.

The procedure for the **PSNR** based **quadtree segmentation** starts by segmenting the image into **blocks** of 32x32 pixels. A neural network model which has only **one** hidden layer node is fitted to the block. The block is reconstructed using the optimum **parameters**. The PSNR between the original and reconstructed blocks is computed. This PSNR is compared to some predefined **PSNR** threshold. The values of this threshold for our work is set to 28 dB and 29 dB for the test images Lenna and Pepper, respectively. This
threshold controls the number of 32x32 blocks in the final segmented image. To ensure the quality of the reconstructed image (in the mean square sense), the value of this threshold for blocks of 32x32 is large. Lower threshold for blocks of 32x32 pixels increases the number of 32x32 blocks in the final segmented image and causes some blockiness in the reconstructed image. If the PSNR of the reconstructed 32x32 block is lower than the predefined segmenting threshold, the block is segmented to four 16x16 blocks. This process continues until the smallest allowable block size is reached. The value of PSNR threshold for block size of 16x16 is selected to be 24 dB for both images Lenna and Pepper. Since the compression ratio for the blocks smaller than 8x8 pixels is not significant, our segmentation algorithm stops splitting the blocks after this point.

Once the image is segmented, neural network modeling and the theory of projection pursuits is invoked to construct a distinct code for each block. Projection pursuits is an efficient iterative function approximation method. This method is a powerful image coding procedure which adds a sense of multiresolution to the coding procedure. To find the optimum parameters for each block, two different approaches are implemented. The first approach is based on the backpropagation (steepest descent) algorithm. The second approach is based on the Gams-Minos (reduced gradient) algorithm. In the first approach, a three-layer neural network is considered for each block. This network is a simple neural network model which has only one node in its hidden layer. Then, the backpropagation algorithm is used to train the network. The inputs are normalized to values between zero and one. The initial weights and biases of the network are initialized to small values between +1 and -1. The number of training iterations is set to 500 epochs and is independent of block size. It should be noted that the sizes of the input and output vector are 2x1 and 1x1, respectively, and are independent of block size.
After designing the three layer network, backpropagation learning is used to train the network until the optimal parameters for each block are obtained. The projection pursuit image coding with PSNR based quadtree segmentation algorithm starts with specifying a global threshold for the PSNR of the reconstructed image and two PSNR segmentation thresholds for the block sizes of 32x32 and 16x16. In this work, the desired global thresholds for the test images Lenna and Pepper are 30 and 31 dB, respectively. Initially quadtree segmentation algorithm segments the image into blocks of 32x32. Then each block is trained with a simple neural network which has only one hidden layer node. After training, the optimum weights and biases are obtained. Using the input pixel values and optimum weights and biases, the block is reconstructed. The PSNR between the original and reconstructed block is computed. This threshold is compared to quadtree segmentation PSNR threshold for block size of 32x32. In this work, this threshold is set to 29 dB and 30 dB for the test images Lenna and Pepper, respectively. If the reconstructed block has a PSNR lower than this threshold, the block is segmented to four 16x16 blocks and the same procedure is repeated. Otherwise, the difference between the original pixel values and the reconstructed ones is computed. This error block is treated as an input for another network which has only one hidden layer node. After training, the error block is reconstructed using the optimum weights and biases. The reconstructed image at this stage is the reconstructed error block added to the reconstructed block from the previous stage. The new error block is again obtained and treated as the input for the next stage of projection pursuit coding. This process is repeated until the overall PSNR is larger than the global PSNR or the desired bit rate is achieved. The final reconstructed image should have a PSNR which is close to the specified PSNR. The image coding procedure with PSNR based quadtree segmentation, neural networks, and projection pursuits is shown in Figure 3.7.
The image coding procedure with variance based quadtree segmentation, neural networks, and projection pursuits is shown in Figure 3.8. This algorithm starts with specifying a global threshold for the PSNR of the reconstructed image and two variance segmentation thresholds for the blocks of size 32x32 and 16x16. The image is segmented to blocks of 32x32 pixels. The variance of each block is computed. If the variance of 32x32 block is larger than the segmentation threshold, a neural network with one hidden layer node is fitted to the block. The block is reconstructed and the PSNR between the original and the reconstructed image is computed. If this PSNR is larger than the global PSNR, the algorithm stops. Otherwise, another neural network with one hidden layer node is fitted to the residual block and the process is repeated until the bit-rate or global PSNR is satisfied. If the variance of 32x32 block is smaller than the segmentation threshold, the 32x32 block is segmented to four 16x16 blocks and the same procedure repeated for each segmented block.

The two controlling parameters in this algorithm are the global PSNR and the number of stages generated. The first parameter controls the overall PSNR of the reconstructed image. The second parameter controls the bit-rate required for each block. In turn, it should be evident that the blocks extracted from smooth parts of the image are "simple" to code as compared to blocks which are heavily detailed. The simple blocks can then be processed by a small number of stages, whereas larger number of stages are required by more complicated blocks. The number of parameters which are sent to the decoder for each block is fixed and is independent of the block size. The more detailed the blocks are, the larger the number of stages, and hence the bit rate is higher. The maximum number of stages allowed in this work is set to eight.
The second approach for finding the optimum parameters is based on the Gams-Minos (reduced gradient method). In this case, the \textit{quadtree} segmentation process is the same as in the first approach. The algorithm is invoked to code each block using the optimum parameters which are obtained from the optimization process of Gams-Minos. The procedure in this case exactly follows the first approach.

Simulations have been carried out to demonstrate the effectiveness of the proposed algorithm. The reconstructed test image \textit{Lenna} with variance based \textit{quadtree} segmentation at a bit rate of .14 bit per pixel (bpp) is shown in Figure 3.9. The PSNR of the encoded image is 29.11 dB. The reconstructed test image Lenna with PSNR based \textit{quadtree} segmentation at a bit rate of .14 bpp is shown in Figure 3.10. The PSNR of the encoded image is 30 dB. For comparison, the JPEG algorithm was used to encode the test image Lenna at the bit rate of .14 bpp. The JPEG decoded image has a PSNR of 27.47 dB and is shown in Figure 3.11. The reconstructed test image Pepper with variance based \textit{quadtree} segmentation at a bit rate of .16 bpp is shown in Figure 3.12. The PSNR of the encoded image is 30.17 dB. The reconstructed test image Pepper with PSNR based \textit{quadtree} segmentation at bit rate of .14 bpp is shown in Figure 3.13. The PSNR of the encoded image is 30.22 dB. The JPEG encoded image at a bit rate of .14 bpp is shown in Figure 3.14. The PSNR of JPEG decoded image is 21.62. The reconstructed images with the proposed algorithm are superior to JPEG decoded images both in terms of PSNR and the subjective quality. The blockiness artifact of JPEG decoded images are very obvious. The PSNR based \textit{quadtree} segmentation has an improvement of almost 1 dB over the variance based \textit{quadtree} segmentation method. By analyzing the two segmentation results, it can be seen that variance is not an optimum criterion for some blocks. The PSNR based segmentation and coding can more effectively allocate the bits in blocks such that the quality of the reconstructed blocks in PSNR is insured. So overall it can be concluded that
the quadtree algorithm with PSNR measure can more effectively isolate the homogeneous blocks from more detailed ones. A summary of coding results for images Lenna and Pepper are given in Table 3.1 and 3.2, respectively.

Table. 3.1. The PSNR and bit-rate of the encoded test image Lenna with the three algorithms, variance based algorithm, PSNR based algorithm and JPEG.

<table>
<thead>
<tr>
<th>Test Image Lenna</th>
<th>PSNR dB</th>
<th>Bit-Rate bpp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variance Based Algorithm</td>
<td>29.11</td>
<td>1.4</td>
</tr>
<tr>
<td>PSNR Based Algorithm</td>
<td>30</td>
<td>.14</td>
</tr>
<tr>
<td>JPEG</td>
<td>27.47</td>
<td>.14</td>
</tr>
</tbody>
</table>

Table. 3.2. The PSNR and bit-rate of the encoded test image Pepper with the three algorithms, variance based algorithm, PSNR based algorithm and JPEG.

<table>
<thead>
<tr>
<th>Test Image Pepper</th>
<th>PSNR dB</th>
<th>Bit-Rate bpp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variance based Projection Pursuits</td>
<td>30.17</td>
<td>.16</td>
</tr>
<tr>
<td>PSNR Ratio Based Projection Pursuits</td>
<td>30.22</td>
<td>.14</td>
</tr>
<tr>
<td>JPEG</td>
<td>21.62</td>
<td>.14</td>
</tr>
</tbody>
</table>
Select a global PSNR threshold and two PSNR segmentation thresholds for the block sizes of 32x32 and 16x16

Segment the image into blocks of 32x32.

Fit a simple neural network

Reconstruct the block and find the PSNR of the reconstructed block

The PSNR of the reconstructed block is less than the global PSNR?

Yes

No

stop

The PSNR of the reconstructed block is less than the PSNR segmentation threshold for block size of 32x32?

Yes

Segment the block into 4, 16x16 blocks and repeat the same steps which is performed for 32x32 blocks.

No

Fit another neural network with one hidden layer node and repeat this process until the required bit rate or required PSNR is satisfied.

Figure 3.7. The image coding procedure with PSNR based quadtree segmentation, neural networks, and projection pursuits.
Select a global PSNR threshold and two variance segmentation thresholds for the blocks of size 32x32 and 16x16.

Segment the image into blocks of 32x32.

Find the variance of the block.

The variance is less than the variance threshold for blocks of 32x32?

Segment to 4, 16x16 blocks.

Fit a simple neural network.

Repeat the same steps as for 32x32 blocks.

Reconstruct the block and find the PSNR of the reconstructed block.

The PSNR of the reconstructed block is less than the global PSNR?

No

Repeat with the residual image block.

Yes

Stop.

Figure 3.8. The image coding procedure with variance based quadtree segmentation, neural networks, and projection pursuits.
3.6. Conclusions

A novel approach for low bit rate image compression is presented in this chapter. It has been demonstrated that quadtree based image segmentation can be an effective and efficient mechanism for isolating blocks of distinct perceptual significance. Our quadtree segmentation algorithm divides the image into variable-size blocks based on one of two possible "measures of activities" within the block. The measures of activity used in this chapter are image variance and PSNR of the reconstructed block. It is shown that quadtree algorithm with PSNR measure can isolate the homogenous blocks from the more detailed ones more effectively than the variance based algorithm.

A new variable-rate coding technique is introduced which is based on neural network modeling together with the projection pursuits algorithm. This is similar to parallel, self-organizing neural networks with continuous inputs and outputs. It is shown that this algorithm can adaptively construct a better approximation for each block until the desired PSNR or bit rate is achieved. This method also finds adaptively the optimum network configuration and is superior to the proposed algorithm in chapter two in that respect. Perceptually, we have obtained high quality image reproduction at rates of .14 bit per pixel. Objective measure of performance using PSNR, while of moderately limited use in assessing quality, has nevertheless given values superior to the JPEG decoded images.

In summary, this work has introduced a new direction of research on model based image compression using neural networks, projection pursuits, and quadtree segmentation.
Figure 3.5. Test image Lenna.
Figure 3.6. The 512x512 test image Pepper.
Figure 3.9. The encoded test image Lenna with variance based quadtree segmentation.
Figure 3.10. The encoded test image Lenna with PSNR based quadtree segmentation
Figure 3.11. The JPEG encoded test image *Lenna* at bit rate of .14 bpp and PSNR of 27 dB.
Figure 3.12. The encoded test image Pepper with variance based quadtree segmentation.
Figure 3.13. The encoded test image Pepper with PSNR based quadtree segmentation.
Figure 3.14. The JPEG encoded test image Pepper at a bit-rate of 0.14 and PSNR of 21.62.
4.1. Introduction

Digital color images are usually represented by three bands, corresponding to red, green, and blue tristimulus values. Since there is often substantial correlation between the color planes, independent processing of each plane is not optimal. The traditional approach in color image processing is to apply a coordinate transformation to the RGB image to obtain the luminance and chrominance components [29]. A luminance-chrominance coordinate space is preferred because the image components are then relatively uncorrelated. The chrominance images usually have less spatial bandwidth than the luminance image. Therefore, the chrominance images are usually subsampled by two both horizontally and vertically. Monochrome image processing techniques can then be applied to the luminance and subsampled chrominance images. Finally, an inverse coordinate transformation is applied to reconstruct the image.

In this chapter, a novel approach for low bit rate color image coding is introduced. In this new technique, image compression is achieved first by segmenting the luminance and chrominance images into regions of different sizes based on the FSNR of the reconstructed block and then constructing a distinct code for each block by neural network modeling together with projection pursuits. The process is terminated when the desired PSNR or bit rate is achieved.
The organization of this chapter is as follows: Section 4.2 is devoted to human visual system. Section 4.3 addresses NTSC color transformation. Section 4.4 presents the PSNR based quadtree segmentation of luminance and chrominance images. Section 4.5 is devoted to color image coding by neural network modeling and projection pursuits. The experimental results are provided in Section 4.6. Section 4.7 presents conclusions.

4.2. Human Visual System

We present in this section a functional description of the human eye as a background for constructing a model of visual processing consistent with physiology [44], [30], [27], [20]. The human visual system is a part of the nervous system and is managed by the powerful computer, the brain. Light from an external object is focused by the cornea and lens to form an image of the object on the retina at the back of the eye ball. The retina is the neurosensory layer of the eye and its area is about 12.5 cm. It transforms the incoming light into electrical signals that are transmitted to the visual cortex through the optic nerve. The retina consists of a layer of photoreceptors and connecting nerve cells. The receptors contain photo sensitive pigments that are capable of absorbing light and initiating the neural response. Human eye has two kind of photoreceptors: rods and cones. A normal eye contains about 130 million rods and 6.5 million cones. In the region surrounding the fovea, only cones are present and they are densely packed. The density of cones decreases rapidly as we move away from the fovea, whereas the density of rods increases. Rods and cones are different enough to be examined separately. Cones are responsible for spatial acuity and color vision at normal day light level (photopic vision). Rods are sensitive to shapes and provide low light vision (scotopic vision) but do not contribute to color vision. At light levels between photopic and scotopic range, both cones and rods provide vision. The distribution of cones in retina is highest in the vicinity of the optical axis of the eye. That is why a precised detail vision is obtained only when
the eye is "fixed" on the image, in other words, when the image is formed at the fovea. In this region, there are about 120 cones per degree which limits the visual resolution to one minute of arc. Light absorbed by the receptors leads to chemical reactions that bleach the photo sensitive pigment, which reduces the light-sensitivity in proportion to the fraction of pigment bleached. A change in ambient illumination causes the amount of bleached pigment to rise or fall to a new equilibrium level, and this provides a mechanism for adopting to different light levels.

Light is electromagnetic energy that can be completely specified at a point in the image plane by its wavelength distribution. Not all electromagnetic radiation is visible to the human eye. In fact, the entire light portion of the electromagnetic spectrum is only within the narrow wavelength region of 380 to 780 nanometers. The radiation incident on the eye produces two sensations. The first one is the brightness sensation which is produced by aggregate of various wavelengths. The second one is the visual sensation due to which an observer distinguishes between different spectral compositions. The human retina contains three different color receptors (cones) that are sensitive to three overlapping areas of the visible spectrum. The three types of receptors have sensitivity peaks at approximately 445 (called blue), 535 (called green) and 570 (called red) nanometers. Each type of receptors integrate the energy in the incident light at various wavelengths in proportion to their sensitivity for that wavelength. The three responses have considerable overlap, a feature that is necessary to allow the visual system to distinguish light at different wavelengths. If, for example, wavelengths in the range 540-570 nanometer excited only one of the three cone types, the visual system could not distinguish between intensity differences and wavelength differences in this range.
4.3. NTSC Color Transformation

There is considerable correlation between the red, green and blue components of natural color images. This is due to the fact that most natural pictures do not contain large areas with saturated colors. To exploit these correlations, a rotation of the coordinate axes can be performed to transform the \([R, G, B]\) color vector so that the new components are uncorrelated. The transformation generally aims at packing a large fraction of spectral energy into a small number of transformed spectral planes, so the majority of the transform planes would contain little energy and can be encoded at very low bit rates.

The optimum transform in terms of maximum energy compaction is the Karhunen-Loeve Transform (KLT), which unfortunately is image-dependent [29]. To avoid the computational complexity associated with the KLT, an image independent transform with reasonable decorrelation capability can be chosen.

4.3.1. Color Coordinate Systems and Transforms

Most colors can be matched by a mixture of three suitable chosen primary colors. Given three defined matching stimuli \(R, G\) and \(B\), which could, for instance, be our three monochromatic lights of wavelengths 700 nanometer (red), 546.1 nanometer (green) and 438 nanometer (blue), the amounts of these three stimuli needed to match any color enable it to be related systematically to all other colors. Thus the equation:

\[ KC = R_c R + G_c G + B_c B \]

represents \(K\) units of color \(C\). Now the amount of color \(C\) can be regarded as a physical or photometric quality. Instead of specifying a color by its tristimulus values \(R, G\) and \(B\), colorimetrists often use normalized quantities called chromaticity coordinates [20], [27]. These are expressed as:
Using the chromaticity coordinates, the color $C$ can be represented as:

$$\frac{KC}{R_c + G_c + B_c} = rR + gG + bB$$

Of course, since $r + g + b = 1$, two chromaticity coordinates are sufficient.

In 1931, the international body of color scientists called the CIE (Commission Internationale de L'Eclairage) defined three new stimuli $X,Y$ and $Z$ in terms of which standard-observer results could be expressed [27]. It is possible to calculate the amounts of $X,Y$ and $Z$ needed to match any color from the amounts of $R,G$ and $B$ of a red, green, blue system, provided that transformation equation relating the two systems are known. First we have

$$C(C) = R(R) + G(G) + B(B)$$

Since we know that

$$R = l_1X + l_2Y + l_3Z$$
$$G = l_4X + l_5Y + l_6Z$$
$$B = l_7X + l_8Y + l_9Z$$
By substituting for \((R),(G),(B)\), we obtain:

\[
C(C) = V_1(X) + V_2(Y) + V_3(Z)
\]

where

\[
V_1 = l_1R + l_4G + l_7B
\]
\[
V_2 = l_2R + l_5G + l_8B
\]
\[
V_3 = l_3R + l_6G + l_9B
\]

Two properties of the CIE coordinate system make it an interesting and useful choice [20]. First, the \(Y\) tristimulus value corresponds to the definition of luminance. It therefore follow:: that all the luminance of the test color has to be balanced by the \(Y\) stimulus. Thus, variations of the amounts of \(X\) and \(Z\) affect the color of the match, but leave any difference in luminance unchanged. Also unlike an \(R\), \(G\), \(B\) system, where sometimes certain tristimulus values have to be negative for a match, the tristimulus values in the CIE-XYZ system are always positive. Chromaticity coordinates can also be defined in CIE-XYZ system. For the tristimulus values \(X\), \(Y\), \(Z\) the chromaticity coordinates are given by [27]

\[
x = \frac{X}{X+Y+Z}
\]
\[
y = \frac{Y}{X+Y+Z}
\]
\[
z = \frac{Z}{X+Y+Z}
\]

since \(x+y+z=1\), a color can be specified by the two chromaticity coordinates \(x\) and \(y\).

The ability of the eye to see fine detail depends for the most parts on differences in luminance in the pattern and only to a much smaller extent on color contrast [20]. This suggests that if the information in a color image could be divided into its luminance
content: and its color content, then only the luminance information need to be transmitted at high bit-rate, and bandwidth could be saved by transmitting the color information at reduced rate.

As an example, in television camera, three electrical signals, $E_R$, $E_G$, $E_B$ (usually expressed as voltages) are obtained that are proportional at each point of the picture to its red, green and blue contents. The luminance $L$, at any point in the picture will be denoted by:

$$L = L_R R + L_G G + L_B B$$

where $L_R$, $L_G$, $L_B$ are the luminance of the units in which the red, green, blue contents are measured. It is therefore possible to produce an electrical signal $E_L$, proportional to luminance $L$, by adding together the same proportion of the signals $E_R$, $E_G$, $E_B$:

$$E_L = L_R E_R + L_G E_G + L_B E_B$$

Instead of transmitting the three signals $E_R$, $E_G$, and $E_B$, $E_L$ and two of the other signals, say $E_R$ and $E_B$ are transmitted. The signal $E_L$ could be transmitted with broad bandwidth, and the signals $E_R$ and $E_B$ with narrow bandwidth. At the receiver, the $E_G$ signal can be recovered by following operation:

$$E_G = \frac{1}{L_G} E_L - \frac{L_R}{L_G} E_R - \frac{L_B}{L_G} E_B$$

There are some advantages in transmitting a separate high definition luminance signal [20]. First if the $E_R$ and $E_B$ signals suffer from interference, they will not affect the luminance displayed. This has a beneficial effect, because the human eye is more
sensitive to luminance changes than to chromaticity changes. The second advantage is compatibility with monochrome images. In order to produce monochromatic version of color images, the only thing the receiver has to do is to ignore the color signals.

Greater advantages can be obtained if, instead of transmitting the signals $E_L, E_R, E_B$, the luminance signal is accompanied by two color-differences or chrominance signals, such as $E_R - E_L$ and $E_B - E_L$ [20]. The receiver then recovers the signal $E_G - E_L$ by the following operation:

$$E_G - E_L = \frac{1 \cdot L_R - L_G - L_B}{L_G} E_L - \frac{L_R(E_R - E_L)}{L_G} - \frac{L_B(E_B - E_L)}{L_G}$$

A new luminance signal $E_Y$ (the suffix $Y$ indicates the $Y$ of the CIE $X, Y, Z$ system) can be denoted as:

$$E_Y = lE_R + mE_G + nE_B$$

where

$$l = \frac{L_R}{L_R + L_G + L_B} \quad m = \frac{L_G}{L_R + L_G + L_B} \quad n = \frac{L_B}{L_R + L_G + L_B}$$

so that $l + m + n = 1$.

$E_Y$ is still a true measure of luminance but is now expressed in units $L_R + L_G + L_B$ times as large as those used for $E_L$. The color difference signal now becomes $E_R - E_Y$ and $E_B - E_Y$. The receiver can recover $E_G - E_Y$ by the following operation:

$$E_Y = lE_R + mE_G + nE_B$$

Substituting for $l$, $m$ and $n$ in the expression above, we get
\[ E_Y = \frac{L_R}{L_R + L_G + L_B} E_R + \frac{L_G}{L_R + L_G + L_B} E_G + \frac{L_B}{L_R + L_G + L_B} E_B \]

\[ E_Y (L_R + L_G + L_B) = L_R E_R + L_G E_G + L_B E_B \]

\[ E_G - E_Y = -\frac{L_R}{L_G} (E_R - E_Y) - \frac{L_B}{L_G} (E_B - E_Y) \]

\[ E_G - E_Y = -\frac{1}{m} (E_R - E_Y) - \frac{n}{m} (E_B - E_Y) \]

Therefore, the three transmitted signals are the luminance \( E_Y \), and the two color-difference signals, \( E_R - E_Y \) and \( E_B - E_Y \). At the receiver, the signals \( E_R, E_G, E_B \) are obtained by adding \( E_Y \) to the difference-signals. The use of color difference signals of this type has further advantages if it is further arranged that the relative sensitivities of the three channels of the camera are arranged such that for whites, grays, and blacks, \( E_R = E_G = E_B \). Since \( l + m + n = 1 \), and \( E_Y = l E_R + m E_G + n E_B \), it follows that for whites, grays, and blacks, \( E_R = E_G = E_B = E_Y \) [20]. Therefore, the color-difference signals \( E_R - E_Y \) and \( E_B - E_Y \) are both zero. This reduces the need for transmitting information additional to that contained by the \( E_Y \) signal for most scenes consisting of fairly low color saturation.

4.3.2. The NTSC Color Transformation

The National Television System Committee (NTSC) in 1952 defined the color television system that is currently in use in North America and Japan [27]. The standard color primaries in terms of CIE XYZ chromaticity coordinates are as follows [20]:

\[
\begin{array}{ccc}
 x & y & z \\
 R: & .67 & .33 & .00 \\
\end{array}
\]
In the previous section, a new luminance signal $E_Y$ was introduced. $E_Y$ is a true measure of luminance and is given by $E_Y = lE_R + mE_G + nE_B$ where $l+m+n=1$. The luminance $L_X$ and $L_Z$ of unit quantities of $X$ and $Z$ are both equal to zero and so the luminance $L_Y$ of unit quantity of $Y$ may be set arbitrarily equal to unity. With the amounts of $R$, $G$ and $B$ measured in luminance units, a fact we indicate by using the symbols $R_L, G_L, B_L$, we may therefore write

\[
\begin{align*}
0.33R_L &= 0.67X + 0.33Y + 0.00Z \\
0.71G_L &= 0.21X + 0.71Y + 0.08Z \\
0.08B_L &= 0.14X + 0.08Y + 0.78Z 
\end{align*}
\]

To obtain the luminance signal $E_Y$ in the NTSC system, the values of $l, m,$ and $n$ are needed. The coordinate system which is used to solve for these parameters are the chromiiticity coordinate system ($x, y$, and $z$). For standard illumination $C$ ($S_C$), $E_R = E_G = E_B$ and hence the corresponding amounts of $R$, $G$ and $B$ light are equal to $S_C$. Hence we can write:

\[ S_C = 0.333R + 0.333G + 0.333B \]

when the amounts of $R$, $G$, $B$, and $S_C$ are measured in luminance units, the above equation becomes

\[ 0.333lR_L + 0.333mG_L + 0.333nB_L = 0.333(l + m + n)S_C \]
but \(1+m+n=1\), therefore

\[
S_{Cl} = lR_L + mG_L + nB_L
\]

Substituting for \(R_L, G_L, B_L\) results in

\[
S_{Cl} = (.67/\cdot.33)X + lY + (.00l/\cdot.33)Z + \quad .21m/\cdot.71)X + mY + (.08m/\cdot.71)Z + \quad (.14n/\cdot.08)X + nY + (.08n/\cdot.08)Z
\]

\(S_C\) can also be written in luminance units namely

\[
.316S_{Cl} = .310X + .316Y + .374Z
\]

\[
S_{Cl} = (.310/\cdot.316)X + 1.0Y + (.374/\cdot.316)Z
\]

By comparing the last two expressions for \(S_{Cl}\), we obtain:

\[
.67l/\cdot.33 + .21m/\cdot.71 + .14n/\cdot.08 = .310/\cdot.316
\]

\[
.00l/\cdot.33 + .08m/\cdot.71 + .78n/\cdot.08 = .374/\cdot.316
\]

\[l + m + n = 1\]

Solving the three simultaneous equations for \(l, m\) and \(n\), we obtain

\[l = .299 \quad m = .587 \quad n = .114\]

Hence, the luminance signal \(E_Y\) in NTSC can be represented as \([20], [27]\):

\[
E_Y = .299E_R + .587E_G + .114E_B
\]
Having designated the luminance, we must now define two chrominance values in order to specify the three-dimensional color transformation. The two chrominance signals, $E_R - E_Y$ and $E_B - E_Y$ are first reduced in amplitude by factors of $1.14$ and $2.03$ respectively. Then, a $33$ degree of axis rotation is performed to obtain the two chrominance signals $E_I$ and $E_Q$:

$$E_I = \frac{E_R - E_Y \cos 33^\circ - E_B - E_Y \sin 33^\circ}{1.14}$$
$$E_Q = \frac{E_R - E_Y \sin 33^\circ + E_B - E_Y \cos 33^\circ}{1.14}$$

In matrix form, these results are expressed as [20]

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} .299 & .587 & .114 \\ .596 & -.275 & -.321 \\ .212 & -.523 & .311 \end{bmatrix} \begin{bmatrix} E_R \\ E_G \\ E_B \end{bmatrix}$$

### 4.4. **Quadtree Segmentation of Color Images**

The quadtree segmentation of color images is accomplished by segmenting each color component separately. First the R, G, B images go under NTSC transformation to produce Y, I, Q images. The goal of this transformation is to decorrelate the original R, G and B images of size $N \times N$. This decorrelation generally results in the signal energy being redistributed mostly in the Y component. Since Q and I components contain a small percentage of total energy, they can be decimated by two to create a new $N/2 \times N/2$, Q and I components. The three components $Y (N \times N), Q (N/2 \times N/2)$ and $I (N/2 \times N/2)$ are the input images to the quadtree segmentation algorithm.
Figure 4.1. **Quadtree** segmentation map for the 512x512 luminance image **Lenna**.
Figure 4.2. Quadtree segmentation map for 256x256 the chrominance image $I$ for Lenna.
Figure 4.3. Segmentation map for the 256x256 chrominance image Q for Lenna.
The process of selecting quadtree segmentation thresholds for luminance and chrominance images are the same. The measure of activity used to distinguish the smooth blocks from the heavily detailed ones is the PSNR of the reconstructed block. In this case, first a global PSNR threshold for the reconstructed image is selected. Then, depending on the block size, a set of block thresholds are chosen such that the reconstructed block with PSNR lower than these thresholds are split to smaller blocks. The larger is the block size, the closer is the block threshold to the global threshold. It is intended to set the block threshold close to the global threshold for the block size of 32x32 pixels. This reduces the blockiness artifact which may be introduced by encoding of two many blocks of 32x32 pixels.

The result of quadtree segmentation of luminance Y image is almost the same as the previous section for grayscale Lenna. The segmentation of chrominance images I and Q are very simple. Since these images are very smooth, the segmentation of these images mostly resulted in blocks of 32x32 pixels. The PSNR based quadtree algorithm can effectively isolate the homogeneous regions of the three components of color image from the heavily detailed ones. The quadtree segmentation map for luminance image Y and color components I and Q are shown in Figures 4.1, 4.2, and 4.3 respectively.

4.5. Color Image Coding by Neural Networks and Projection Pursuits

Once the components of color image (Y, I, Q) have been quadtree segmented, each block is coded by using neural network and projection pursuit. This is a multiresolution coding technique which improves the quality of the reconstructed block at each step. In the first step of the iteration, the current image is the original block, and in step k, the current image is the residual or the error image that is the difference between the original block and linear combination of all the (k-1) previous approximations. Projection pursuit progressively approximates the image until the predefined thresholds are satisfied. To
control the bit rate and the quality of the reconstructed block, two thresholds have been defined. The bit rate is controlled by the number of stages which the construction progresses. The quality of the reconstructed image (in mean squared sense) is controlled by the desired PSNR. The blocks extracted from the smooth part of the image require a small number of stages compared to the blocks which are heavily detailed.

Backpropagation learning rule is used to train the network for every block in the Y, I, and Q images. Below we outline the algorithm for the luminance Y. The training procedure for each block in luminance Y starts with a simple network which has only one hidden node. After training, the optimum weights and biases are obtained. Using the input pixel values and optimum weights and biases, the block is reconstructed. The signal to noise ratio between the reconstructed block and the original block is computed. If this value is more than some predefined threshold, the training stops. Otherwise, the differences between the original pixel values and reconstructed ones are computed. This error image is treated as an input for another network which has only one hidden node. After training, the same procedure is repeated. The approximation process is terminated when the overall error drops below the desired threshold or the desired bit-rate is achieved.

4.5. Experimental Results

Simulation is carried out to demonstrate the effectiveness of the proposed algorithm. The 512x512 RGB color test image Lenna is shown in Figure 4.5. The first step in the coding process is the NTSC color transformation. The NTSC color transformation is performed on the [R, G, B] color vectors to produce a new set of vectors which are almost uncorrelated. The result of this linear transformation is a luminance image Y, and two chrominance images I and Q. The human eye is relatively insensitive to chrominance transitions in high detail areas of the image. Therefore, the chrominance images I and Q
do not require the full resolution for subjectively acceptable picture quality in most cases. In order to reduce the resolution of chrominance images, a decimation process is performed on I and Q images. The I and Q images are subsampled by two vertically and horizontally to produce a low resolution chrominance images of size 256x256.

After transformation, the next step in the coding process is quadtree segmentation of the image. Quadtree algorithm segments the image into regions of different sizes based on some measure of "activity" in each region. An analysis of test images (Y, I, Q) showed that the blocks larger than 32x32 pixels are almost inhomogeneous. Therefore, our quadtree segmentation algorithm starts with a partitioning of the image into blocks of 32x32 pixels. The measure of activity used in this work is based on the PSNR of the reconstructed block. The smallest block size which is allowed in this work is 8x8 pixels.

The result of quadtree segmentation of luminance image is the same as in the previous chapter for grayscale image Lenna. The chrominance images are very smooth. The quadtree segmentation of chrominance images produces mostly blocks of 32x32 pixels. The number of blocks resulted from quadtree segmentation of images Y, I, and Q for different block sizes is shown in Table 4.1.

<table>
<thead>
<tr>
<th>Images</th>
<th>32x32</th>
<th>16x16</th>
<th>8x8</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>116</td>
<td>394</td>
<td>664</td>
<td>1174</td>
</tr>
<tr>
<td>I</td>
<td>60</td>
<td>16</td>
<td>0</td>
<td>76</td>
</tr>
<tr>
<td>Q</td>
<td>63</td>
<td>4</td>
<td>0</td>
<td>67</td>
</tr>
</tbody>
</table>

Table 4.1. The number of blocks resulted from quadtree segmentation of images Y, I, and Q for different block sizes for the test image Lenna.
Once the image is segmented, the theory of neural networks and projection pursuits is invoked to construct a distinct code for each block. To find the optimum parameters of the model, first a three layer neural networks is designed for each block. Then, the backpropagation learning rule is used to train the network for every block in the Y, I, and Q images. The training procedure for each block in luminance Y starts with specifying a global threshold for the PSNR of the reconstructed image and two PSNR segmentation thresholds for the block sizes of 32x32 and 16x16. In this work, the desired global threshold for the test image Lenna is 31 dB. Initially quadtree segmentation algorithm segments the image into blocks of 32x32. Then each block is trained with a simple neural network which has only one hidden layer node. After training, the optimum weights and biases are obtained. Using the input pixel values and optimum weights and biases, the block is reconstructed. The PSNR between the original and reconstructed block is computed. This threshold is compared to quadtree segmentation PSNR threshold for block size of 32x32. In this work, this threshold is set to 29 dB for the test image Lenna. If the reconstructed block has a PSNR lower than this threshold, the block is segmented to four 16x16 blocks and the same procedure is repeated. Otherwise, the difference between the original pixel values and the reconstructed ones is computed. This error block is treated as an input for another network which has only one hidden layer node.

Table 4.1. The bit-rate of the encoded test image Lenna with the two algorithms, PSNR based algorithm, and JPEG.

<table>
<thead>
<tr>
<th>512X512 test image Lenna</th>
<th>Bit-Rate (bpp)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR Based Algorithm</td>
<td>.14</td>
</tr>
<tr>
<td>JPEG</td>
<td>.22</td>
</tr>
</tbody>
</table>
After training, the error block is reconstructed using the optimum weights and biases. The reconstructed image at this stage is the reconstructed error block added to the reconstructed block from the previous stage. The new error block is again obtained and treated as the input for the next stage of projection pursuit coding. This process is repeated until the overall PSNR is larger than the global PSNR or the desired bit rate is achieved. The final reconstructed image should have a PSNR which is close to the specified PSNR. The coding of the chrominance images (I, Q) follows the same procedure as the coding of luminance image. The image coding procedure with PSNR based quadtree segmentation, neural networks, and projection pursuits is shown in Figure 4.4.

The two controlling parameter in this algorithm are the global PSNR and the number of stages generated. The first parameter controls the overall signal to noise ratio of the reconstructed image. The second parameter controls the bit rate which is desired for each block. In turn, it should be evident that the block extracted from smooth parts of the image are "simple" to code with respect to block which are heavily detailed. The simple blocks can then elaborated by a small number of iterations, whereas a higher number of iteration are required by more complicated blocks. The number of parameter which are sent to the decoder for each block is fixed and is independent of the block size. The more detailed the blocks are, the larger the number of stages, and hence the bit rate is higher.

The maximum number of allowed stages in this work is set to six.

Simulation have been carried out to demonstrate the effectiveness of the proposed algorithm. The coded color test image Lenna with signal to noise based quadtree segmentation at a bit rate of .14 bit per pixel (bpp) is shown in Figure 4.6. For comparison, the JPEG algorithm was used to encode our test image at the bit rate of .22 bpp. Blockiness artifact presented at the JPEG decoded image can be seen in Figure
4.7. The decoded images with the proposed algorithm are superior to JPEG decoded image both in terms of bit rate and the subjective quality.

Do the NTSC color transformation on RGB images to obtain YIQ images.

- The I and Q images are subsampled by two both horizontally and vertically
- Follow the coding procedure presented in Chapter 3 for the decimated images I and Q.
- Interpolate the images I and Q to their original sizes.
- For the luminance image Y, follow the coding procedure presented in Chapter 3.

Figure 4.4. The color image coding procedure with NTSC color transformation, PSNR based quadtree segmentation, neural networks, and projection pursuits.
4.6. Conclusions

A novel approach for low bit rate color image compression is presented in this chapter. It is shown that the NTSC color transformation can effectively decorrelate the R, G, and B images. This decorrelation allows the luminance, Y, and chrominance Q and I to be coded at different bit rates without considerable loss of subjective quality. It has been demonstrated that quadtree based image segmentation can be an effective and efficient mechanism for isolating blocks of distinct perceptual significance. Our quadtree segmentation algorithm divides the image into variable-size blocks based on signal-to-noise ratio of the reconstructed block as the measure of activity. The luminance image , Y, and chrominance images, I and Q, are segmented to variable block sizes so that the "simpler" blocks can be isolated and coded with less number of bits.

A new variable-rate coding technique is introduced which is based on neural network modeling together with the projection pursuit algorithm. It is shown that this coding algorithm can adaptively construct a better approximation for each block until a desired PSNR or bit-rate is achieved. For a specified desired PSNR, this method adaptively finds the optimum bit allocations for the luminance and chrominance images. Perceptually, we have obtained high quality image reproduction at rates of 1.6 bit per pixel. Objective measure of performance using PSNR, while of moderately limited use in assessing quality, has nevertheless given values superior to the JPEG decoded images.

In summary, this work has introduced a new direction of research on model based color image compression using neural networks and projection pursuits. Due to the parallel nature of the algorithm, coding of color components can be easily implemented in parallel machines.
Figure 4.5. The 512x512 RGB image Lenna.
Figure 4.6. The JPEG encoded color image at bit-rate of .22 bpp.
Figure 4.7. The encoded color test image Lenna with the proposed algorithm at a bit-rate of .14 bpp.
CHAPTER 5. CLASSIFICATION ACCURACY IMPROVEMENT OF NEURAL NETWORK CLASSIFIERS BY USING UNLABELED DATA

5.1. Introduction

Within the last decade, advances in space and computer technology have made it possible for remote sensing sensors to collect data of higher spatial resolution in wider coverage area, and with an increasing number of spectral bands. Progress in sensor technology for earth observing remote sensing caused the development of high spectral resolution multispectral imaging sensors. The spectral resolution of a remote sensing imaging system refers to the number and widths of the spectral bands that are collected. HIRIS (High Resolution Image Spectrometer), for example, gathers data simultaneously in 192 spectral bands in the 0.4 - 2.5 \( \mu \)m in wavelength region and 30 m spatial resolution\[4],[33\]. AVIRIS (Airborne Visible and Infrared Imaging Spectrometer) covers the 0.4-2.5 \( \mu \)m wavelength region in 224 spectral bands. One objective of using such high-dimensional data is to discriminate among more ground cover classes (e.g., sand, trees, mountain, water, ....) and hence obtain a more detailed understanding about the nature of the materials that cover the surface of the earth.

For classification of remote sensing data many techniques have been utilized over the years. Conventional classification methods, such as maximum likelihood or Bayesian methods, which have been applied successfully in remote sensing in the past, can not be easily applied in classification of very-high-dimensional remote sensing data. These conventional multivariate statistical approaches are based on the assumption of having
nonsingular (invertible) class-specific covariance matrices for all classes. However, this assumption is not valid for very-high-dimensional data especially when there are not enough independent samples in the training set. In order to overcome this shortcoming of statistical models, nonparametric approaches can be applied.

There has recently been considerable interest in using artificial neural networks to classify remote sensing data [2], [3], [38]. The ability of learning in neural networks provides an interesting alternative to the conventional classification methods. In remote sensing data classification, neural network models exploit the following features [2]:

(1) Neural network models do not require any a priori knowledge of the class statistical distribution in data sources. This is a remarkable feature since there is no requirement of explicitly modeling the data in each source.

(2) The neural network approaches avoid the problem in statistical multisource analysis of specifying how much influence each source should have on classification. This implies that the neural network approach becomes more preferable for multisource remote sensing data classification.

In order to design an accurate classifier, training samples should represent all classes of interest. In some applications, the training samples may not be a good representation of the entire class, especially when the samples are collected from spatially adjacent regions. For remote sensing applications, class information must be gathered either by visual inspection of the scene around the same time the data are being collected, or by using an expert analyst for identifying the class labels of data based on their spectral responses. In either case, the number of training samples which can be obtained are usually limited.
The purpose of this paper is to explore and study some techniques for improving the classification accuracy of neural network classifiers by utilizing unclassified observations that may be available in large numbers and with no extra cost. We refer to these unclassified samples as unlabeled data, vs labeled data in which the underlying classes are known. Shahshahani and Landgrebe [33] have shown the effect of additional unlabeled samples in enhancing the classification performance of statistical classifiers.

Including unlabeled observations in the process of designing a neural network classifier has the following potential advantages [33]:

1) The classification accuracy of the classifier can be improved without changing the characteristics of the network.
2) The unlabeled samples may provide information about the classes for which there are few or no training samples available.
3) Training of the network with large number of unlabeled data reduces the effect of small training sample size.

The organization of this chapter is as follows: In Section 5.2, application of neural networks in pattern recognition and remote sensing is presented. In Section 5.3, backpropagation algorithm is briefly discussed. Section 5.4 is devoted to the description of how additional unlabeled samples can be used to improve the classification performance. In Section 5.5, the Colorado data set is presented as a 10 class data from the Colorado region. In Section 5.6, the experimental results are presented. The conclusions are given in Section 5.7.
5.2. Neural Network Classifiers

Artificial neural network classifiers have been demonstrated to be an attractive alternative to conventional statistical classifiers for analyzing remote sensing data. A neural network is an interconnection of basic computational units, each of which is referred to as a "neuron". A neuron can be modeled as a set of synaptic weights, one for every input, plus a bias weight, a summer, and a nonlinear function referred to as the activation function as shown in Figure 5.1. Each unit computes the weighted sum of the inputs plus the bias weight and passes this sum through an activation function to calculate the output value as

\[ y_j = f\left( \sum_i w_{ji} x_i + q_j \right) \]

where \( x_i \) is the \( i \)th input value, and \( w_{ji} \) is the corresponding synaptic weight. The activation function \( f(\bullet) \) maps the range of the weighted sum to a limited range, typically \([0, 1]\) or alternatively \([-1, 1]\). In a multilayer configuration, the network can have more than one hidden layer. In this case, the outputs of the units in one layer form the inputs to the next layer. The inputs to the first layer are the network inputs, and outputs of the last layer are the network outputs. The three-layer neural network which is used in this work is presented in Figure 5.2. Each output node is dedicated to one class. With the Colorado data set described in section 5.5, these are water (\( y_1 \)), Colorado blue spruce (\( y_2 \)), mountane/subalpine meadow (\( y_3 \)), aspen (\( y_4 \)), ponderosa pine (\( y_5 \)), ponderosa pine/douglas fir (\( y_6 \)), engelmann spruce (\( y_7 \)), douglas firlwhite fir (\( y_8 \)), douglas fir/ponderosa pinelaspern (\( y_9 \)), douglas firlwhite firlaspern (\( y_{10} \)). The desired output
representation is such that $y_i$ equals 1 and $y_j$ equals 0 for $j \neq i$ when the input vector belongs to class i.

Figure 5.1. Model of a neuron

Figure 5.1. Model of a neuron
5.3. The Backpropagation Algorithm

In early 1960's, Widrow and Hoff developed the delta rule [16]. It is a supervised learning approach for a two-layer neural network with linear activation functions. This learning rule can be used to calculate changes in the weights and biases so as to minimize the sum squared error of the network. The delta rule uses the method of steepest descent to minimize square error. The delta rule can be implemented to discriminate linearly separable data.

Backpropagation was created by generalizing the delta rule to multiple layer networks and nonlinear differentiable transfer functions. It can be used to discriminate data that are not linearly separable. As in the delta rule, backpropagation involves adjusting the weights and biases of the network so as to minimize the sum squared error of the network. This is done by changing the values of the network weights and biases in the direction of steepest descent with respect to error. Backpropagation training may lead to a local minimum rather than a global one. In some cases the obtained local minimum may be satisfactory, but if it is not, a network with a different structure or different initial weights can be tried.

5.4. Effect Of Additional Unlabeled Observations

In this section, the effect of unlabeled samples in the classification process of parametric classifiers is discussed [33]. Let us consider a classification problem involving m classes with prior probabilities $p_i$ and probability density functions $f_i(x)$. By $e^*$ we denote the Bayes error achieved by using the MAP classifier when $p_i$ and $f_i(x)$ are known. Let $\theta$ denote the vector of parameters of the MAP classifier. Also let $\theta^*$ denote
the true value of $\theta$. The error achieved by using $\theta^*$ in the decision mle is $e^*$, the Bayes error. Now let us assume there exists an estimate of $\theta^*$ denoted by $\hat{\theta}$. If the deviation of $\hat{\theta}$ from $\theta^*$ is not large, one can approximate the error corresponding to the decision rule obtained with $\hat{\theta}$ by using a Taylor series expansion of up to second term:

$$
\hat{\varepsilon} = e(\hat{\theta}) \equiv e^* + [\nabla e(\theta^*)]^T(\hat{\theta} - \theta^*) + \frac{1}{2} ((\hat{\theta} - \theta^*)^T H(e(\theta^*))(\hat{\theta} - \theta^*))
$$

where $\nabla e(\theta^*)$ and $H(e(\theta^*))$ are the gradient and the Hessian matrix evaluated at $\theta^*$, respectively. Using property of trace, $\hat{\varepsilon}$ can be written as

$$
\hat{\varepsilon} = e(\hat{\theta}) \equiv e^* + [\nabla e(\theta^*)]^T(\hat{\theta} - \theta^*) + \frac{1}{2} \text{tr}\{ H(e(\theta^*)) (\hat{\theta} - \theta^*)(\hat{\theta} - \theta^*)^T \}
$$

We next compute the expected error $E[\hat{\varepsilon}]$

$$
E[\hat{\varepsilon}] = e^* + [\nabla e(\theta^*)]^T E(\hat{\theta} - \theta^*) + \frac{1}{2} \text{tr}\{ H(e(\theta^*)) E(\hat{\theta} - \theta^*)(\hat{\theta} - \theta^*)^T \}
$$

$\nabla e(\theta^*)$ is the null vector since $\theta^*$ is an extreme point of $e(\theta)$. If the bias of $\hat{\theta}$ is zero or negligible ($E(\hat{\theta}) = \theta^*$), then the expected value of $\hat{\varepsilon}$ can be approximated as follows

$$
E[\hat{\varepsilon}] \equiv e^* + \frac{1}{2} \text{tr}\left\{ \frac{\partial^2 e(\theta)}{\partial \theta^2} \bigg|_{\theta = \theta^*} \text{COV}(\hat{\theta}) \right\}
$$

where $H(e(\theta^*))$ is denoted by

$$
H(e(\theta^*)) = \frac{\partial^2 e(\theta)}{\partial \theta^2} \bigg|_{\theta = \theta^*}
$$

and $\text{COV}(\theta)$ is defined by
\[ \text{COV}(\hat{\theta}) = E\left[(\theta - \hat{\theta})(\theta - \hat{\theta})^T\right] \]

(6)

Notice that the bias term on the right hand of equation above is non-negative, since it is the trace of the product of two positive semi-definite matrices. Now consider another unbiased estimate, \( \tilde{\theta} \) and also assume \( \text{cov}(\tilde{s}) \leq \text{cov}(\theta) \) (i.e. \( \text{cov}(\tilde{\theta}) - \text{cov}(\hat{\theta}) \) is positive semi-definite). Then, one can show that

\[
\text{tr}\left\{ \frac{\partial^2 e(\theta)}{\partial \theta^2} \mid_{\theta = \theta^*} \text{COV}(\hat{\theta}) \right\} \leq \text{tr}\left\{ \frac{\partial^2 e(\theta)}{\partial \theta^2} \mid_{\theta = \theta^*} \text{COV}(\tilde{\theta}) \right\}
\]

(7)

The above inequality is true because both the covariance matrix and the Hessian matrix at \( \theta^* \) is positive semi-definite (the Hessian matrix is positive semi-definite at \( \theta^* \) since \( \hat{s} \) is a minimum of \( e(s) \), and \( e(s) \) is convex around \( \hat{s} \)). Therefore one can write

\[
\text{tr}\left\{ \frac{\partial^2 e(\theta)}{\partial \theta^2} \mid_{\theta = \theta^*} \text{COV}(\tilde{\theta}) \right\} - \text{tr}\left\{ \frac{\partial^2 e(\theta)}{\partial \theta^2} \mid_{\theta = \theta^*} \text{COV}(\hat{\theta}) \right\} =
\]

(8)

\[
\text{tr}\left\{ \frac{\partial^2 e(\theta)}{\partial \theta^2} \mid_{\theta = \theta^*}[\text{COV}(\tilde{s}) - \text{COV}(\hat{s})] \right\} \geq 0
\]

Therefore, the expected error due to using \( \tilde{\theta} \) in the decision rule is less than the expected error due to using \( \hat{\theta} \):

\[
E[\tilde{\varepsilon}] \leq E[\hat{\varepsilon}]
\]

(9)
Eq. (8) indicates that the covariance of an estimator is a good measure of performance of that estimator. If covariance matrix for estimators $\tilde{\theta}$ and $\hat{\theta}$ satisfy the condition $\text{COV}(\hat{\theta}) \leq \text{COV}(\tilde{\theta})$ (or $\text{COV}(\hat{\theta}) - \text{COV}(\tilde{\theta})$ is positive semi-definite matrix), then the estimator $\hat{\theta}$ is superior to $\tilde{\theta}$ estimator. So overall, an estimator with smallest error covariance matrix is desirable. The Fisher information matrix provide a lower bound on the error covariance matrix of an unbiased estimators. The Fisher information matrix corresponding to a probability density function $f(x)$ can be denoted as:

$$I = \mathbb{E}\left\{ \left[ \frac{\partial}{\partial \theta} \log f(x) \right] \left[ \frac{\partial}{\partial \theta} \log f(x) \right]^T \right\} - \mathbb{E}\left\{ \frac{\partial}{\partial \theta} \left[ \frac{\partial}{\partial \theta} \log f(x) \right]^T \right\}$$

(10)

Let us assume that $\tilde{\theta}$ is an estimate of $\theta^*$ obtained by using only the training samples. Furthermore assume that $\hat{\theta}$ is asymptotically unbiased and efficient. In other words, for large sample sizes, $\mathbb{E}\{ \hat{\theta} \} = \theta^*$ and $\text{cov}(\hat{\theta}) = I_s^{-1}$, where $I_s$ is the Fisher information matrix. The subscript "s" denotes that the Fisher information matrix corresponds to a supervised estimate obtained by using training samples that are drawn from each class separately. The covariance of the estimation error is bounded from below by the inverse of Fisher information matrix. Let us assume that $\tilde{\theta}$ is another estimate of $\theta^*$ obtained by using some unlabeled samples in addition to the training samples. The unlabeled samples are drawn randomly from the mixture of the m classes. If $\tilde{\theta}$ possesses the same properties of asymptotic unbiasedness and efficiency, one can approximate $\text{cov}(\tilde{\theta})$ by $I_c^{-1}$ where $I_c$ is the Fisher information matrix corresponding to
the estimate that is obtained by combining training and unlabeled samples. Provided that unlabeled and training samples are independent, one can write

$$I_c = I_u + I_s$$  \hspace{1cm} (11)

where $I_u$ is another information matrix corresponding to the information contained in the unlabeled samples for estimating $\theta^*$. Since all the information matrices are positive definite, one can write

$$I_c \geq I_s$$  \hspace{1cm} (12)

Therefore, $\text{cov}(\tilde{\theta}) \leq \text{cov}(\hat{\theta})$. Using this result together with Eq. (8), one can conclude that the expected error of the decision rule that uses $\bar{y}$ is less than the error that is obtained by using $\hat{\theta}$. In other words, including the unlabeled samples along with training samples in decision process can reduce the expected error. In the case of neural network classifier, Ruck et. al. [32] showed that the multilayer perceptron trained with the backpropagation algorithm provides a minimum mean squared-error approximation to the Bayes optimal discriminant functions for both the two-class and multiclass problems. So including unlabeled samples in the decision process can reduce the expected error for the neural network classifier as well.

5.5. The Colorado Data Set

The neural network classification method using unlabeled data is used to classify the Colorado data set consisting of the following [3]:
1) **Landsat** MSS data (4 data channels)

2) Elevation data (in 10 m contour intervals, 1 data channel)

3) **Slope** data (0-90 degrees in 1 degree increments, 1 data channel)

4) **Aspect** data (1-180 degrees in 1 degree increments, 1 data channel)

Each channel comprises an image of 135 rows and 131 columns; all channels are co-registered.

The area used for classification is a mountainous area in Colorado. The area has 10 ground cover classes which are listed in Table 1. One class is water; the others are forest type classes. It is very difficult to distinguish between the forest types using the **Landsat** MSS data alone since the forest classes showed very similar spectral responses. With the help of elevation, slope and aspect data, they can be better distinguished.

<table>
<thead>
<tr>
<th>class #</th>
<th>Information Class</th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>water</td>
<td>408</td>
<td>195</td>
</tr>
<tr>
<td>2</td>
<td>Colorado blue spruce</td>
<td>88</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>mountane/subalpine meadow</td>
<td>45</td>
<td>42</td>
</tr>
<tr>
<td>4</td>
<td>aspen</td>
<td>75</td>
<td>65</td>
</tr>
<tr>
<td>5</td>
<td>Ponderosa pine</td>
<td>105</td>
<td>139</td>
</tr>
<tr>
<td>6</td>
<td>Ponderosa pine/Douglas fir</td>
<td>126</td>
<td>188</td>
</tr>
<tr>
<td>7</td>
<td>Engelmann spruce</td>
<td>224</td>
<td>70</td>
</tr>
<tr>
<td>8</td>
<td>Douglas fir/white fir</td>
<td>32</td>
<td>44</td>
</tr>
<tr>
<td>9</td>
<td>Douglas fir/Ponderosa pine/aspen</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>10</td>
<td>Douglas fir/white fir/aspen</td>
<td>60</td>
<td>39</td>
</tr>
</tbody>
</table>

|        | Total                     | 1188     | 831     |

Table 5.1: Training and testing samples for each class on Colorado data
Ground reference data were compiled for the area by comparing a cartographic map to a color composite of the Landsat data and also to a line printer output of each Landsat channel. The training and testing data are obtained by selecting 2019 ground reference points (11.4% of the area). Ground reference consisted of two or more homogeneous fields in the imagery for each class. Overall 1181 pixels were used for training, and 831 pixels were used for testing the classifiers.

5.6. Experimental Results

Simulation is carried out to demonstrate the effectiveness of the proposed method. The neural network architecture which is used for this classification problem is shown in Figure 5.2. The network has three layers (input layer, hidden layer, output layer). In our design, the number of hidden layers has been set to one. The number of neurons in the input layer is seven. The number of nodes in the hidden layer is set to 14. The network is trained with backpropagation learning algorithm to find the optimum weights and biases. The non-linear transfer function in the output layer is chosen as the log-sigmoid function. The tan-sigmoid function is selected to represent the non-linear transfer function in the hidden layer. Both of the transfer functions satisfy the differentiability conditions required by the backpropagation learning rule. The input data are normalized to values between 0 and 1. The weights and biases are initialized to small values between +1 and -1. Since the 10 class Colorado data set is considered in this experiment, the desired output for each input vector is a 10x1 vector. The desired output vector has only one non-zero entry of "1" at the location which matches the class of an input vector.

In the first part of this experiment, supervised learning was investigated. In this process, the network is trained only with labeled (training) samples. The labeled observations are
presented to the network and the training is terminated when the desired error or specified maximum number of iterations is achieved. After obtaining the optimum weights and biases, the classification accuracy of designed neural network classifier is tested. For doing so, the testing samples are presented to the network.

![Figure 5.3. Supervised-unsupervised learning procedure](image)

The actual value of the network for every input is calculated and the output node with maximum value denotes the result of the classification. This experiment is repeated for different number of iterations and different initializations. An average classification accuracy of 52% is obtained for testing samples.
In the second part of this experiment, unsupervised learning is presented together with supervised learning to improve the classification accuracy of the neural network classifier. The network parameters remain the same as in the first part of this experiment. The procedure of supervised-unsupervised training is shown in Figure 5.3. The process of supervised-unsupervised learning starts with training of the network with labeled data for 500 epochs. The obtained weights and biases at 500th epoch is used to initialize the weights and biases for training with the unlabeled samples. 500 samples were selected randomly from the population of the testing data for this purpose. Since the testing data are unlabeled, the information about which class they belong to does
not exist. The class information of unlabeled data is obtained by presenting the selected testing samples to the network and calculating the actual value of the output at every node. The output node with maximum value denote the desired output. Using the obtained desired outputs, initial weights and biases, the network is again trained with selected testing data for 500 epochs. This process of supervised-unsupervised learning is repeated for 75 iterations. At every iteration, the classification accuracy of the testing data is measured.

After supervised training, the testing data was presented to the network, and the ratio of correctly classified samples to the total number of testing samples was obtained at each iteration. The perclass classification accuracy of testing data are shown in Figure 5.4. By analyzing the performance of classifier for class one and seven, it is evident that the network can discriminate these classes almost perfectly. This is not surprising, since the class one (water) and class seven (Engelmaan Spruce) have the most number of training samples in the training set. On the other hand, classes eight and nine had the worst performance since they have the least number of training samples in the training set. Including the testing samples from under represented classes in the training set, improves the classification accuracy of these classes during the supervised-unsupervised training process. Classification accuracy of the network is tested at every iteration. The classification accuracy for a typical seed is shown in Figure 5.5. The sum squared error for a typical seed is shown in Figure 5.6. The average classification accuracy for 3 different seeds (initialization of the random number generator) is shown in Figure 5.7. The average sum squared error for 3 different seeds is shown in Figure 5.8. It is observed that a considerable improvement in classification accuracy is achieved.
Figure 5.5. Classification accuracy of the neural network with supervised-unsupervised learning for a typical seed.
Figure 5.6. The sum squared error of the neural network classifier with supervised-unsupervised learning for a typical seed.
Figure 5.7. Average classification accuracy of neural network classifier with supervised-unsupervised learning for three different seeds.
Figure 5.8. Average sum squared error of neural network classification with supervised-unsupervised learning for three different seeds.
5.7. Conclusions

A new procedure for classification accuracy improvement of neural network classifiers is presented. A large number of classes of interest in remote sensing necessitate a large number of samples to be obtained for training. These training samples are often very expensive and time consuming to obtain. In order to improve the classification accuracy without increasing the number of training data, it is suggested to train the network with the unlabeled testing samples together with the labeled training samples. It is shown that supervised-unsupervised training of neural network classifier can effectively improve the classification accuracy. In particular, including the unlabeled samples from under represented classes in the training set, improve the classification accuracy of these classes during training. It is also shown for the case of parametric classifiers that the expected error of the decision rule that uses labeled and unlabeled samples is less than the one that uses training samples only. Since the output of a neural network such as a backpropagation network approximates the a posteriori probabilities, the same result is obtained for neural network classifiers.
CHAPTER 6. SUMMARY AND CONCLUDING REMARKS

6.1. Summary of Results

In this thesis, two applications of neural networks is investigated. The first one is the low bit rate image compression by using neural networks and projection pursuits. The second one is classification accuracy improvement of neural network classifiers using unlabeled data. The two applications demonstrate that neural networks can be effectively used for image modeling as well as data classification.

In chapter two, a novel approach for low bit rate model-based image compression is presented. It has been demonstrated that quadtree based image segmentation can be an effective and efficient mechanism for identifying blocks of distinct perceptual significance and thereby allowing different coding strategies that are perceptually suited to the individual segmented region. We introduced a new variable-rate coding technique which is based on the amount of activity in each individual block. Central to our coding approach is the use of the variable-rate coding that conceptually corresponds to the idea of designing neural networks with different hidden layer nodes. In this approach, the "simpler" blocks are required a neural network with a small number of hidden layer nodes. On the other hand, the more detailed blocks require a network with large number of hidden layer nodes. Perceptually, we have obtained high quality image reproduction at rates of .25 bpp. Objective measurements of performance using PSNR has nevertheless given values comparable to the JPEG decoded image. The subjective quality of the encoded image with proposed algorithm is also compatible to the JPEG encoded image.

In chapter three, a new algorithm for low bit rate image coding is presented. Image compression is achieved first by segmenting the image into blocks of different sizes based
on two measure of activity, and then constructing a distinct code for each block by invoking the theory of projection pursuits. The two measures of activity which are used in this work are block variance and signal to noise ratio of the reconstructed block. It is shown that the projection pursuits coding algorithm can adaptively construct a better approximation for each block until the desired signal to noise ratio or bit rate is achieved. This algorithm also finds adaptively the optimum network configuration and is superior to the proposed algorithm in chapter two. Objective measure of performance using PSNR has given values superior to the JPEG decoded images. The encoded test image Lenna with the proposed algorithm has a PSNR of 30 dB and bit rate of .14 bpp. The encoded test image Pepper with the proposed algorithm has a PSNR of 30.22 dB and bit rate of .14 bpp.

In chapter four, the projection pursuit image coding is extended to color images. It is shown that the NTSC color transformation can effectively decorrelate the R, G, and B images. This decorrelation allows the luminance image, Y, and chrominance images Q and I to be coded at different bit rate without considerable loss of subjective quality. The segmentation of luminance and chrominance images is done by signal to noise ratio based quadtree algorithm. After segmentation, the projection pursuits algorithm is invoked to encode each block. The encoded test image Lenna with the proposed algorithm has a bit rate of .14 bpp. For comparison, the test image is encoded with JPEG compression algorithm. The JPEG decoded image has a bit rate of .22 bpp. The subjective quality of the encoded color image with proposed algorithm is superior to the JPEG decoded color image.

In chapter five, classification accuracy improvement of neural network classifiers using unlabeled testing data is presented. In order to fully utilize the information contained in high dimensional data, training samples are needed from all classes. In order to increase classification accuracy without increasing the number of training samples, the network
makes use of testing data along with training data for learning. However, the testing data are unlabeled whereas the training data are labeled. It was shown previously for the case of parametric classifiers that decision rules which use both labeled (training) and unlabeled (testing;) samples have a lower expected error than those which use labeled samples only. Since the output of a neural network such as backpropagation network approximates the a posteriori probabilities, the same result applies to neural network classifiers. It is shown that including unlabeled samples from under-represented classes in the training set improves the classification accuracy of some of the classes during supervised-unsupervised learning.

6.2. Future Research

Some future research for image coding with projection pursuits and neural networks are as follows:

-In this thesis, a novel algorithm for low bit rate model-based still image compression is presented. The extension of this work to model-based video image compression is an interesting topic. The video compression algorithms exploit the extensive frame-to-frame redundancy which exists in a sequence of images. Because of the significant redundancy between consecutive video frames, most of the information for the current frame can be determined from adjacent frames. For this purpose, block matching algorithm are utilized to estimate the motion of a block of pixels in the present frame in relation to pixels in the previous frame. After motion estimation, the residual pixel values are coded and sent to the receive]:.

The most common coding algorithm which are used in video compression are based on transform coding. The projection pursuits image coding can be a good alternative for coding of the residual pixel values. After motion estimation, the first step in projection pursuits coding of residual image is the quadtree segmentation. The residual image is
quadtree segmented to variable size blocks based on some measure of activity. After segmentation, the projection pursuits coding algorithm can be used to code each block. Projection pursuits progressively approximates the block until the predefined threshold are satisfied.

-In this thesis, the number of epochs which backpropagation algorithm is trained is fixed to 500 epochs. It is interesting to investigate the coding performance for higher number of epochs. As the number of epochs increases, the sum squared error decreases in some cases. This can result in coding performance with higher PSNR without increases the bit-rate.

-In color image coding with projection pursuits and neural networks, the NTSC color transformation is used. The coding performance of the coding algorithm with different color transformations can be investigated.

-In this thesis, quadtree segmentation algorithm is used to segment the image into variable size blocks. The coding performance for different segmentation algorithms can be investigated.

Some future research for improvement of the neural network classifiers using unlabeled data are as follows:

-In the supervised-unsupervised training, the maximum number of epochs which the network is trained is set to 500 epochs. The classification accuracy improvement for higher number of training epochs can be investigated.

-In the unsupervised learning, the class information of the testing data can be obtained by some statistical models (fuzzy neural network approach).

-In this work, the same network is used for supervised and unsupervised training. The classification accuracy of classifiers with different networks for supervised and unsupervised training can be investigated.
REFERENCES


