Neural Network Studies of Lithofacies Classification

David Anthony Harris

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy
to the
University of Edinburgh
1994

Dedication

Among those who have helped bring me to the point of completing this thesis, I especially wish to record my gratitude to my mother, Marjorie Harris, and my grandmother, Marjorie Black. My father, Robert John Harris, died before I began it. I owe much to him. I am also grateful to my fiancée, Caroline King, for her love and support.

I submit this thesis with humility — there is much in it that could be improved — and trust that it will be read in the same spirit:

> A little learning is a dang'rous thing;
> Drink deep, or taste not the Pierian spring:
> There shallow draughts intoxicate the brain,
> And drinking largely sobers us again...
> ...A perfect Judge will read each work of Wit
> With the same spirit that its author writ:
> Survey the Whole, nor seek slight faults to find
> Where nature moves, and rapture warms the mind;
> Nor lose, for that malignant dull delight,
> The gen'rous pleasure to be charm'd with Wit...

- From "The Essay on Criticism" by Alexander Pope (1688-1744).

Abstract

Exploration for hydrocarbons and other resources requires that large amounts of data be interpreted and used to infer the geology of extensive regions. Many different types of data are used. They are interpreted by geologists and sedimentologists in the light of experience.

Artificial neural network models implemented on computers provide a powerful means of performing tasks such as pattern classification. Such tasks are difficult to perform using rule based methods, as we often do not know how to specify appropriate rules.

We show that artificial neural networks can be used to discriminate between images of different lithofacies (types of rocks). This discrimination is based upon textural differences in the rocks, which are quantified by measures of texture derived from the rock images and used as inputs to the network. Neural network performance is good compared to a very simple alternative technique, that of K nearest neighbours.

A particular set of such measures is that based on the grey level cooccurrence method. These measures have interesting properties; in particular their expectation values can be calculated exactly for images generated by exactly solvable Ising models. The measures are themselves probabilities for the joint distribution of pixel values in an image, so that they can be used to generate images in a stochastic process.

Declaration

This thesis has been composed by myself and it has not been submitted in any previous application for a degree. The work reported within was executed by me, unless otherwise stated.

Acknowledgements

I wish to thank my supervisor, David Wallace, for his help and encouragement. Dr Jonathon Lewis of the Department of Petroleum Engineering at Heriot-Watt University also gave invaluable help; I could not have completed this work without either of them.

Image preprocessing was done on a CM-200 parallel computer (supported by Scottish Enterprise and the Advisory Board for the Research Councils) in Edinburgh Parallel Computing Centre.

I also wish to thank the staff of the Graphics Workshop at Edinburgh University for their kind help with scanning the images in Appendix B

Parts of Chapter 4 appeared in a paper presented at EUROCAIPEP '93, King's College, Aberdeen, September 20th - 22nd 1993 with Dr Jonathan Lewis of Heriot-Watt University[70].

I did the work upon which this thesis is based while suported by the Science and Engineering Research Council.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis, work is described whose goal is to discriminate between different types of rocks. The discrimination is based upon differences in the textures of the rocks, which are quantified by functions of the images known as texture measures. These measures are used as inputs to neural network models, which perform the discrimination. The measures themselves have interesting properties.

This Introduction reviews briefly the motivation and aims of the project(§1.1), the justification for using neural networks in this work (§1.2) and the contents of the rest of the thesis (§1.3). It should be read with the flowchart, Figure 1.1.

First, the need for research in this area is established.

1.1 Subsurface Reservoir Description

The work is motivated by the need of the petroleum industry to obtain information from remotely sensed data gathered in the course of exploration. Similar needs arise in other fields, and the techniques required lead to interesting problems of theory (Chapter 5).

Hydrocarbon fluids (oil and gas) are found in underground reservoirs to which

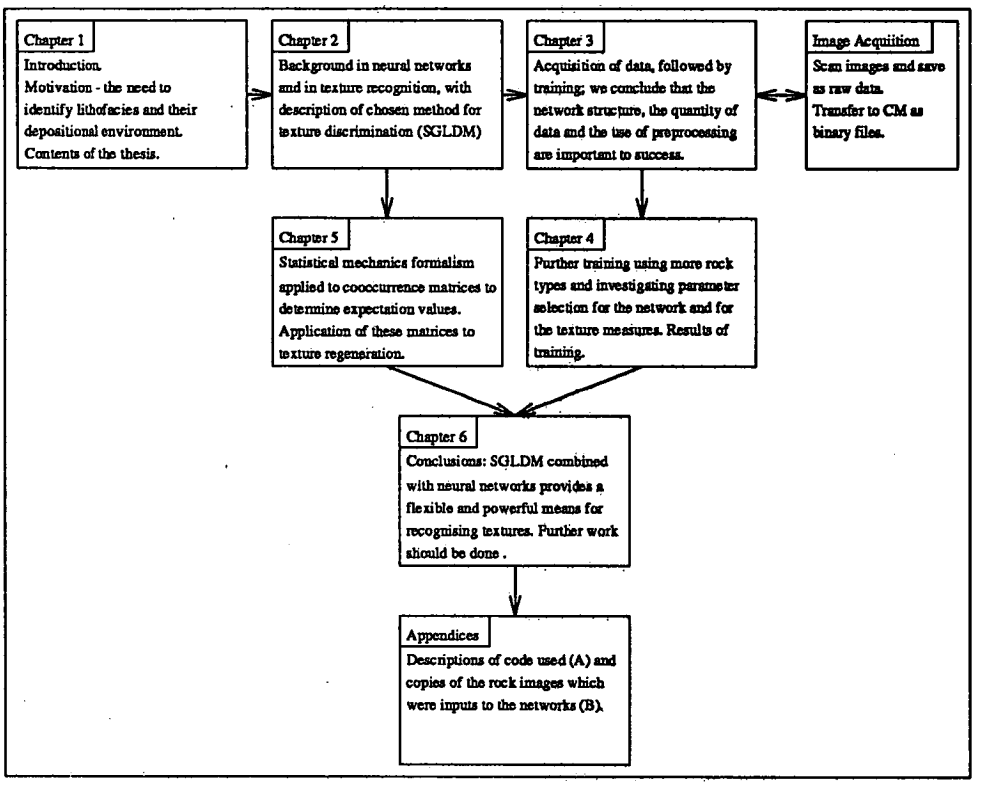

Figure 1.1. Flowchart to illustrate contents of the thesis. For each chapter, the principal contents and conclusions (where appropriate) are given. The arrows show which chapters are closely related. See the text for more detail.

they have migrated through porous rock, or through faults in impermeable rock, from the strata where they were formed. The reservoirs are formations which are able to contain these fluids; for instance, domes of impermeable rock under which oil and gas rise and are trapped.

In developing an oil field, it is necessary to gain an understanding of the depositional environment of the lithofacies. The depositional environment is the environment obtaining at the time and place that the strata in question were laid down — the broadest categories are continental, coastal shelf or deep marine; each of these has many subclassifications. A lithofacies may be defined[95, p. 120] as

> A body of rock with certain specified attributes that distinguish it from other rock units.

Many properties may the play the part of the "specified attribute" in the foregoing definition; for instance (and relevant to this thesis) the rock **texture** may be used, either from actual core or from an image. Consider a depositional process, such as an estuary system. Sand and silt are carried with the moving water and deposited. Where the river flows quickly, only larger, coarser grains are shed. But where it flows more slowly, fine grains are deposited. The resulting sedimentary rocks will reflect, by their fine or coarse texture, the depositional environment. This is shown in Figure 1.2 which shows a similar, but artificial, difference in the texture formed by two different materials — sand and gravel — on the site of the former Morris works at Cowley near Oxford.

Knowledge of the depositional environment allows sedimentologists to obtain a qualitative understanding of the lithofacies types present. This makes it possible to understand, and model, the flow properties of the region giving an understanding of how the various fluids present (oil, gas and brine) interact with one another and with the rocks in which they are contained. Understanding of these properties is important if geologists are to predict whether or not resources (oil and or gas) are present in economically significant quantities and to determine how their extraction may best be accomplished.

The depositional environments are recognised from core (rock extracted from a well in the process of drilling) and from wireline log data. Wireline logs are records made, after drilling, of various measurements obtained from instruments carried with the drill. Examples of quantities commonly measured by such instruments are neutron backscatter, gamma ray activity, the speed of sound in the rock, and its density. These measurements are characteristic of different lithofacies types.

Obtaining core requires that drilling be repeatedly stopped, and the core

removed. This is relatively expensive and slow, compared to simply drilling, so it is not practical to do it for every well. This means that the volume of rock obtained by coring is very small compared to the volume of rock being explored. This being so, other information is often used to infer lithofacies types. Especially relevant to the present research, wireline logs have been used[128] as inputs to a neural network for this purpose.

Borehole images can be obtained without coring and so they can be provided cheaply in larger (and more statistically significant) quantities than can core. Such images give the possibility of better results. They have the speed and cost advantages of wireline logs but a greater information content (if not that of core). Imaging techniques such as **formation microscanning (or microimaging)**[1] are based on taking measurements of the resistivity of the rock face at many points over a 2D field. This is done by pads which travel behind the drill. These values of resistivity can then be assembled into an image. Such techniques give high resolution images, allowing improved identification of lithofacies types.

1.2 Neural Network Models

Neural networks have been used for a variety of pattern recognition and classification tasks, and also for other purposes. Their inspiration was as simplified models of neurons, so they have been used to gain insight into the functioning of the brain. They have also been studied as examples of many body systems in statistical physics. See [89] for a full treatment, [98] for a more general view and [5] for a readable introduction.

These models were originally motivated by the information processing capabilities of the brain. The "processing elements" of the brain, the neurons, are

[1] FMS/ FMI

slow (taking approximately 1ms to switch) compared to transistors, of which modern digital computers are (ultimately) built. Nevertheless, the brain is capable of many tasks which conventional computers are not; such problems as speech recognition and pattern recognition seem easy for the brain but hard for computers.

The brain contains an enormous number of its slow processing elements — perhaps 10^{11} of them — each one connected to many others (one neuron can be connected to a thousand or more others). The state of activity of a neuron — crudely, "firing" or "not firing" — depends upon the sum of the activity coming into it from other neurons. It is believed that this highly parallel, highly connected structure gives the brain its impressive abilities. Artificial neural networks (ANNs) are designed to exploit these advantages, by imitating the biological example upon conventional or parallel computers. This has to be done within the restrictions of the hardware available, limiting the numbers of "neurons" or "units" and the degree of their interconnectivity.

Artificial neural networks thus consist of neurons, con-nected by links or "weights" of various strengths which moderate the signals between them. These weights correspond to the synapses found in the brain. Synapses may be inhibitory, tending to reduce activity, or excitatory. In artificial neural networks, negative and positive weights achieve these functions.

When used for classification, neural networks modify their internal structure (usually, the strengths of their weights; few network models modify their structure by adding or deleting neurons; for some that do see [47]) according to simple rules which allow them to "learn" the classifications of known examples. They can then be used on "unseen" examples, and are thus ideal for use in tasks where it is not clear how a set of rules could be defined. It is this aspect which is made use of in the current work.

1.3 Contents of Thesis

In this thesis, neural networks are applied to the task of classifying high resolution photographs of various lithofacies types, or "pseudo FMS images".

First, in Chapter 2 background material is discussed. In §2.1 neural networks are described, the strengths and weaknesses of some different types of net are discussed, and a comparison is made with traditional statistical methods of pattern classification; the two approaches are not as distinct as they might appear. There follows in §2.2 a discussion of **texture** since this is a key property of the images with which we work. Much work has been done in the field of human texture recognition, and this is drawn upon as a motivation for the particular technique that we used for texture recognition, the **Spatial Grey Level Dependence Method**[30] or **SGLDM**, described in §2.3.

Chapter 3 contains a description of our first attempts to discriminate lithofacies types, using a restricted set of images (§3.2). The importance of using preprocessed data, rather than the raw pixel values, as inputs to the network is emphasised by a comparison (§3.3, §3.4) between work using preprocessed image features and work using raw pixel values. In §3.5 we see that the use of preprocessing allows us to use a particularly simple network architecture (the simple perceptron) and learning algorithm (the Perceptron Learning Rule or PLR) instead of the more sophisticated Multilayer Perceptron (MLP). Finally, in §3.6 we discuss some parameters which are important to the preprocessing. The conclusions follow in §3.7.

Chapter 4 concerns systematic training using a larger set of images (described in §4.1). The training uses a larger network with a more complicated structure, which is necessary since the problem has become considerably more difficult. This training is described in §4.2. The results are discussed in §4.3 and a comparison is made (§4.5) with the results of the K nearest neighbour method, a more

traditional pattern classification technique. §4.6 describes missclassifications of images by the neural network and compares them with the behaviour of humans. Possible future work is also described in §4.7.

In Chapter 5, studies are made of the SGLDM and the cooccurrence matrix (§2.3) in the context of statistical mechanics. First, some basic concepts of statistical mechanics are described (§5.1) and then it is shown that the cooccurrence matrix may be determined for the 1D Ising model (§5.2) and the 2D Ising model (§5.3) from first principles.

Chapter 6 summarises the work and points ahead to possible further work.

Appendix A contains technical details of the hardware and software used, especially the parallel computing aspects.

Appendix B contains examples of the images used for training.

We now turn to the background of neural networks, and to the question of how textures can be described and recognised.

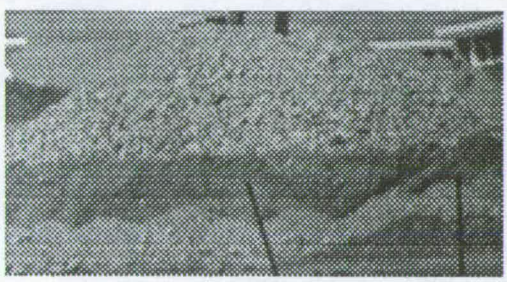

Figure 1.2. A pile of coarse gravel has been deposited on top of a layer of sand. The resulting difference in texture is clear. Similardifferences in the grain size of sediment in water or wind created formations would (ultimately) produce rocks with different textures.

Chapter 2

Techniques for Discrimination

This chapter presents the techniques employed for discrimination, which form the background to the project. Neural networks are described in §2.1 and some of their properties discussed. In §2.2 we look at the nature of texture, at how humans discriminate between different textures, and at measures which have been developed to quantify texture description. Finally, the particular technique used in this research, the spatial grey level dependence method[67], which is based on cooccurrence matrices, is described in detail in §2.3. We begin by considering what neural networks are, and how they work.

2.1 Neural Networks

Neural networks are used in this thesis to classify the images of rocks and so it is important to understand their strengths and limitations. We describe, therefore, a basic neural network architecture (the simple perceptron) and a more sophisticated one (the multilayer perceptron). Both models are used for discrimination in this thesis.

2.1.1 Training and Testing

When using a neural network to perform a task such as pattern classification, we expect to train it to classify previously unseen patterns correctly. We are provided with a set of correctly classified patterns; it is standard procedure to divide these into two sets — a "training set" which is used to train the network, and an independent "test set" which is used to assess how well the trained network performs in classifying previously unseen data. The capability to classify such data correctly is known as the "generalisation ability" of the network.

2.1.2 Simple Perceptron

The simple perceptron (Figure 2.1), or single layer perceptron, is the most elementary neural network architecture. It was introduced in the 1950s by Rosenblatt[122] and is extensively and carefully treated by Minsky and Pappert in [100], the first edition of which was published in 1969. Minsky and Pappert's book has been represented as having stopped neural network research for a decade, and as being unduly pessimistic; the first claim is an exaggeration, and the second is rather harsh — see the postscript in the 1988 edition[100]. The architecture is worth discussing not only in itself but because more complicated networks are often constructed by combining simple perceptrons (henceforth, simply "perceptrons") into multilayer perceptrons. In this thesis, we use the perceptron in Chapter 3, for training which involves only two output categories.

Architecture

The perceptron has N inputs. Its state is defined by an N component weight vector $\mathbf{w}$. Each weight $w_i, i \in [1, N]$ is a connection strength between one of the inputs and the output; to determine the output o^μ for a pattern ξ, the inputs

ξ_i are presented and the scalar product $x = \mathbf{w}.\xi$ calculated. The output of the perceptron is then given by one of several possible functions $f(x)$, for example

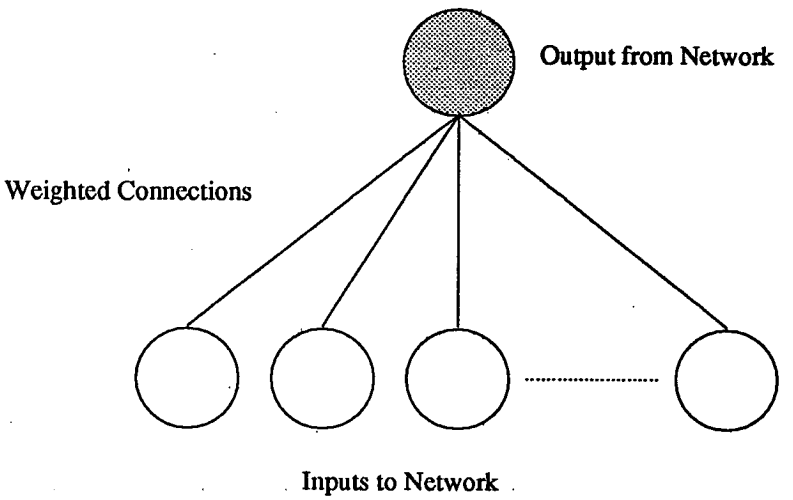

Figure 2.1. Simple Perceptron

$$f(\mathbf{w}.\xi) = \begin{cases} \mathbf{w}.\xi & \text{Linear perceptron} \\ \Theta(\mathbf{w}.\xi) & \text{Binary outputs 0 or 1} \\ 2\Theta(\mathbf{w}.\xi) - 1 & \text{Binary outputs } \pm 1 \end{cases}$$

where

$$\Theta(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

For a binary perceptron, the weights define a hyperplane in the N dimensional space of the patterns, dividing this space into two; input patterns on one side of the plane give an output "on" while those on the other side give an output "off".

Learning Rule

If the network is to store information, it is necessary, given an architecture, to set the weights in some way. This may be done either by some incremental

procedure or by a "one shot" modification. For architectures such as the Hopfield network[89, ch. 2], it is possible to prescribe the weights, for instance by the Hebb rule. For a perceptron (of one or more layers) it is more usual to use an incremental procedure. We present a pattern and adjust the weights to give the desired output. Patterns may be presented one by one and the updates made after each presentation (online learning). Alternatively, the entire training set may be presented, the weight updates Δ_i^μ stored, and their resultant $\sum_\mu \Delta_i^\mu$ used to modify the existing weights (batch learning). Both methods can be effective (see [89, p. 119] for a discussion of this in the context of multilayer nets).

The most commonly used rule for updating the weights, either online or batch, is the Perceptron Learning Rule[122]. We present an N bit pattern $\xi_i^\mu, i \in [1, N]$ with μ a label for the patterns. Then the corresponding weight modification is

$$\Delta_i^\mu = \alpha \left(o^\mu - t^\mu\right) \xi_i^\mu \tag{2.1}$$

where o^μ and t^μ are the actual and desired outputs, respectively, for the pattern μ and α is a constant controlling the learning rate. In practice, an additional weight w_0, the **bias** is often used with an input fixed at 1. The bias frees the hyperplane defined by the weight vector from having to pass through the origin.

The Perceptron Learning Rule (PLR) guarantees that an appropriate set of weights will be found, if such exists[100, 11]. It will fail if the set of input/output pairs is not **linearly separable**, that is, if the two output classes cannot be separated by a hyperplane. Otherwise, some vector $\mathbf{W}$ will be found by the PLR such that $\Theta(\mathbf{W}.\xi^\mu) = o^\mu$ for all μ. It is shown by Minsky and Pappert[100, p. 182] that even when the set of patterns to be learned is nonseparable, use of the PLR still generates a bounded weight vector. In numerical simulations the accuracy with which the weights are stored is finite, so that the learning process has only a finite number of states available. These are not necessarily sampled uniformly, however, so it is not possible to say whether the PLR performs better

or worse than chance in these circumstances. See [47, p. 12] for discussion of this.

Because the perceptron has a simple architecture, theoretical work has been done to investigate its capacity and learning behaviour. There is a difference between absolute capacity, determined by the architecture, and the number of patterns that can actually be learned with a given rule. It has been shown that continuous valued units (linear or nonlinear) have a capacity $P_{MAX} = N$ while threshold units with random continuous inputs have $P_{MAX} = 2N$[29, 102, 54]. The capacity possible in a given realisation of the perceptron depends on the learning rule used. Learning behaviour has been studied, and convergence proved for generalisations of the Perceptron Learning Rule[55].

The behaviour of the PLR for non-linearly separable pattern sets represents a limitation imposed by the architecture, rather than a difficulty with the learning rule. Attempts have been made to ameliorate this limitation of the architecture by devising more subtle learning rules. These are designed to give good, convergent behaviour in the non-separable case without degrading performance on those fractions of the training set which are separable. The perceptron will then classify as many as possible of the patterns in the training set correctly. If most patterns in the training set are linearly separable, it should correctly classify these, only failing on a few rogue examples which are not separable. Similar classifications should be found if the learning procedure is repeated using different random starting weights and/ or a different order of pattern presentation .

Frean[47, 48] introduces the "thermal" perceptron. The weight update Δw_i^μ given by (2.1) is multiplied by $\exp(-\frac{|\mathbf{w}.\xi|}{T})$. The parameter T acts like a temperature (hence "thermal") in that a low T reduces the size of the update step, "freezing" the perceptron. Patterns which are grossly missclassified tend to be ignored in the weight update. The PLR part of the rule ensures that the update is zero for patterns which are already correctly classified. The exponential factor

damps down the size of the update for patterns which lie far from the dividing hyperplane — and which are grossly wrong. This rule is found to perform better than the Pocket algorithm[51] which retains a "good" set of weights[1] (in its pocket) and reverts to them if its performance degrades. It also outperforms the Pocket with the addition of a "ratchet"[53] which improves on Pocket at the cost of further computational effort.

All three of these algorithms succeed fairly well in achieving convergence. However, they cannot overcome the representational limitations of the perceptron architecture. For this, we require a more complicated architecture, such as the multilayer perceptron, which is described in the following section and used in Chapter 4.

2.1.3 The Multilayer Perceptron

This architecture was introduced by Rosenblatt[122],[89, p. 90] at the same time as the "simple" perceptron — it took some time for the nomenclature to settle down. However, a training algorithm was not available to begin with; the back-propagation algorithm which, with its variants and improvements is now widely used, was reinvented at least three times[89] — see[18, 140, 110] — before coming to general attention[98]. Indeed, when the first edition of "Perceptrons", which explored the properties of the simple perceptron, was published in 1969 the authors gave the lack of a training algorithm for the MLP as one of their reasons for pessimism about the field of neural networks in general.

[1] Those which have remained unchanged for longest

Architecture

We restrict ourselves to feedforward networks with connections only between adjacent layers. There are no connections "jumping" layers, and no connections back from higher to lower layers. Both of these features are used in more general neural networks. With these restrictions, the multilayer perceptron (MLP) (Figure 2.2) consists of simple perceptrons arranged in layers with the outputs from one layer going to the inputs of the next one. There must be at least one "hidden" layer; the convention is to neglect the inputs in counting the layers, and to refer to the number of layers of **weights** so that Figure 2.2 shows a two layer network.

The MLP avoids the representational problems associated with the simple perceptron. Instead of partitioning pattern space in two with one hyperplane, it is able to construct more complicated divisions by combining hyperplanes. The increased complexity raises a new problem; what is the most appropriate architecture? If we restrict ourselves to feedforward nets with connections only between adjacent layers, how many layers should there be? And how many units in each layer? Answers exist to the first question. The second is not so straightforward.

Some rigorous results are available for the number of layers required. At most two hidden layers are necessary to represent a particular set of functions to arbitrary accuracy, given enough hidden units[31]. One hidden layer is sufficient to represent any smooth function[32, 74]. Of course, these theoretical bounds may be impractical; the hidden layers may require large numbers of units, and convergence may be very slow. An important issue here is input representation[89, p. 143]. By doing some preprocessing on the data, the problem can be made a great deal easier for the network, and the number of nodes required reduced. This issue is discussed in Chapter 3.

A rule of thumb often quoted is that for generalisation to occur, one requires a number of patterns equal to the product of the number of hidden nodes and the number of inputs. (Alternatively — and equivalently — one requires that the number of patterns be greater than or equal to the number of weights.) If the net is too small, it will be unable to represent the set of patterns; if it is too large, it will overfit the data and generalise poorly.

The simplest procedure for finding the right size of the hidden layer(s) is to guess the size required, constructing the net "by hand" and adjusting it in the light of training performance. Some approaches to network design rely on various criteria to halt training before overgeneralisation begins[103, 139]; for instance, one may retain a third "validation" set of patterns. The network is trained on the training set until performance on the validation set begins to fall off.

Other approaches to this problem use different learning rules, which take into account the network architecture. A weight decay term in the learning rule encourages unnecessary weights to fall away to zero, effectively removing connections. The weights can be examined then, and nodes deleted if they do not perform any useful function[88]. Weight sharing reduces the complexity of the network; the two techniques can be combined[106]. "Constructive" algorithms such as Upstart[47], Cascade Correlation[39] and Tiling[99] (with Tower[52] as special case) build a net to suit the problem.

Learning Rules

There are many ways to adjust the weights in a MLP. We describe below the **backpropagation** algorithm of MacLelland and Rumelhart[98], since this has been widely used and is the basis for the QuickProp scheme used in our training. The basis of the technique is to define an error as a function of the weight values, and then do gradient descent in the space of weights. It is necessary to compute

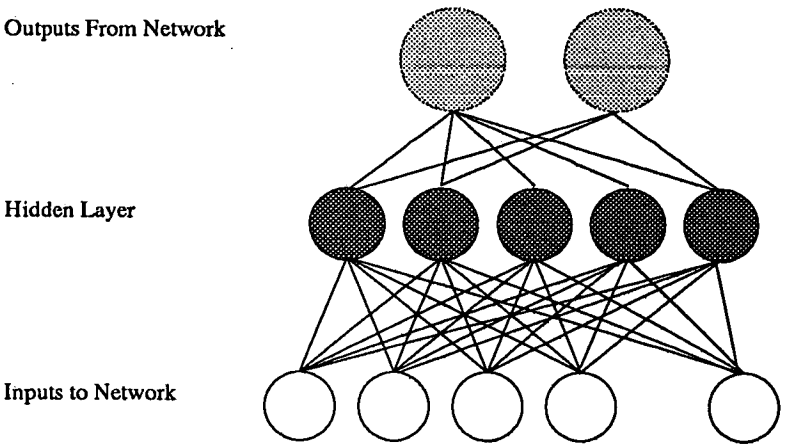

Figure 2.2. Multilayer Perceptron

the partial derivatives of the error function with respect to each weight. The procedure is guaranteed to reach a local minimum, if the steps taken are very small.

Other methods have been proposed that would allow larger steps to be taken. To do this systematically, we need to know the second derivative of the error function. Since this would involve a great deal of computation, approximations to the second derivative are sometimes used. Standard iterative techniques may also be used, such as Newton's method[89, p. 124], or line search methods like steepest descents, including conjugate gradient[89, p. 125].

An interesting, and different, approach is to formulate the problem as a set of differential equations, going from discrete to continuous weights, and then use standard ordinary differential equation (ODE) solving techniques to determine the weights[109, 44].

Backpropagation

The following derivation is based on [98, ch. 2]. See also [89, §6.1].

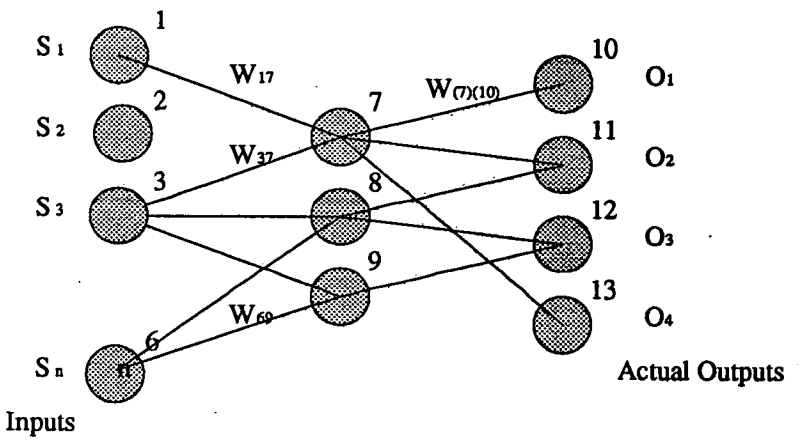

Figure 2.3. Multilayer Perceptron. There are 6 units in the input layer (not all shown), three in the hidden layer and four outputs. Some of the weights are also shown. Each unit is numbered; these numbers correspond to the subscripts in the derivation of the backpropagation algorithm

Let t_j be the jth element of the desired output for a particular pattern, (that is, the target value which an output unit of the network is required to assume) and o_q be the actual output from the qth unit (in the whole network) for the same pattern. s_i is the ith element of the input for the same pattern. (See 2.3.)

w_{qr} is the weight from the qth unit to the rth unit. The labels q and r run over all the units, in every layer, but many of the weights are fixed at zero as we assume that the net is feedforward (only adjacent layers are connected and there are no "backwards" connections).

We label the units U_q. The diagram shows a net with one hidden layer, $n = 6$ inputs and $m = 4$ outputs. The hidden layer has three nodes so there are $n + m + 3 = 13$ nodes in all (not all of the nodes are shownI. Some of the weights are shown and some of these are labelled.

We use a semilinear activation function (the output is a nondecreasing function of the inputs and is differentiable). The activation of the rth unit is

$$a_r = \sum_q w_{qr} o_q \tag{2.2}$$

where the sum is over the units that send inputs to U_r. It could in principle be over all units in the net, but in a feedforward net, is over all the units in the previous layer. Then the output from the same unit is

$$o_r = F_r(a_r) \tag{2.3}$$

where F_r is the transfer function, which could vary from unit to unit; for simplicity we set it to be constant and equal to F.

Now define an error function, given a particular pattern.

$$E = \frac{1}{2}\sum_j (t_j - o_j)^2 \tag{2.4}$$

with the sum over the output units only. We require

$$\Delta w_{qr} \propto -\frac{\partial E}{\partial w_{qr}} \tag{2.5}$$

to be the change in weight w_{qr} caused by the presentation of this particular pattern. Now

$$\frac{\partial E}{\partial w_{qr}} = \frac{\partial E}{\partial a_r}\frac{\partial a_r}{\partial w_{qr}}$$

and from (2.2),

$$\frac{\partial a_r}{\partial w_{qr}} = \frac{\partial}{\partial w_{qr}}\left(\sum_k w_{kr} o_r\right) = o_q$$

Now, **define**

$$\delta_r = -\frac{\partial E}{\partial a_r}$$

so that

$$-\frac{\partial E}{\partial w_{qr}} = \delta_r o_q$$

We need to know δ_r for every unit in the network. Now

$$\delta_r = -\frac{\partial E}{\partial a_r} = -\frac{\partial E}{\partial o_r}\frac{\partial o_r}{\partial a_r}$$

The second term is, from 2.3, simply $F'(a_r)$, the derivative of the transfer function for the rth unit evaluated at the input value a_r. Now consider the first term. There are two cases. First, where U_r is an output unit.

$$\frac{\partial E}{\partial o_r} = \frac{\partial}{\partial o_r}\left(\frac{1}{2}\sum_j (t_j - o_j)^2\right)$$

where, again, j runs over the output units only. Clearly, only the error on U_r contributes. So

$$\frac{\partial E}{\partial o_r} = -(t_r - o_r)$$

giving

$$\delta_r = (t_r - o_r)f'(a_r) \tag{2.6}$$

Secondly, where u_r is not an output unit

$$\frac{\partial E}{\partial o_r} = \sum_k \frac{\partial E}{\partial a_k}\frac{\partial a_k}{\partial o_r}$$

where the sum is over those units which receive outputs from U_r. Substituting for a_k, this gives

$$\frac{\partial E}{\partial o_r} = \sum_k \frac{\partial E}{\partial a_k} w_{rk}$$

and from the definition of δ_r

$$\frac{\partial E}{\partial o_r} = -\sum_k \delta_k w_{rk} \tag{2.7}$$

with

$$\delta_r = F'(a_r)\sum_k \delta_k w_{rk} \tag{2.8}$$

From (2.5) we have $\Delta w_{qr} \propto \delta_r o_q$. This means that the weight from unit q to unit r should be changed by an amount proportional to the product of an error, available at the receiving unit (δ_r) and the output of the sending unit (o_q).

The other equations merely specify this error. It is determined recursively, beginning with the output units. So the algorithm proceeds in this way.

1. Present inputs, calculate the state of the first layer, then the second layer ...the output units layer.

2. For the output units, the error signal is

$$\delta_r = (t_r - o_r)F'(a_r)$$

   which enables the weights leading to the output units to be modified.

3. For the weights on the previous layer, the error signal is given by (2.8). The k summation is over the units in the output layer, with w_{rk} the weights from the last hidden layer to the output layer, whose updates were calculated in the previous step. (Although the modification was calculated then, it is not applied until all the other modifications have been determined.)

4. For the network shown above, there are no more layers of weights to modify. In general, repeat step 3 until the input layer is reached.

The second phase of learning (steps 2 — 4) in which errors propagate backwards, moderated by the weights, is computationally similar in cost to the forward phase. It is this "backpropagation" which gices the algorithm its name. Finally, with all the updates calculated, they can be applied to the weights.

All of the working above assumed — for simplicity — that only one pattern was being presented. In reality a whole set of such patterns are involved, so the

weight modification is

$$\Delta w_{qr}^{p}$$

with p representing the particular pattern.

This procedure for determining the weights is based on gradient descent, and it suffers from the pitfalls which this implies. For instance, there will be many local minima in the error surface. This can lead the network to get trapped in a less than optimal solution, or it may oscillate if the stepsize is finite. Convergence may be very slow, if the error surface is flat along a valley floor with steep sides. In order to address these and other problems numerous variants and modifications have been designed; comparisons between the various methods have to be done with great care, so that like is compared with like (see [37] for a discussion of this issue, and some useful benchmarks).

Variants and adaptations of Backpropagation

The basic backpropagation scheme outlined above has been extensively modified in order to address various problems. There are modified learning rules designed to increase speed by making use of second derivative information — for example Quickprop[37], where a quadratic approximation to the error function is used, so that (where it is safe to do so) the algorithm can go straight to a minimum instead of iterating to it. Self adapting backpropagation[136] varies the update step size to improve convergence time. Alternatively, different step sizes may be employed for every dimension[76, 38]. An additional "momentum" term is usually added to the update step[89, p. 123], to speed convergence and to help prevent the system from being trapped in local minima.

Inputs		Output
+1	+1	−1
+1	−1	+1
−1	+1	+1
−1	−1	−1

Table 2.1. Truth Table for Exclusive-OR Function

2.1.4 Applications of Multilayer Nets

Neural nets have been used for a variety of "real" applications as well as for "toy" problems such as exclusive-OR (XOR). In this problem, the inputs and outputs are the elements of the truth table (see Table 2.1) for the XOR function. The problem is not linearly separable.

The NETtalk project produced a network able to pronounce English (or rather American) text[125]. Neural networks have been used to predict protein secondary sequences from amino acid sequences[117, 12], to perform hyphenation in text[16, 17], recognise sonar targets[58, 59], navigate a vehicle on a winding road[114] and compress and regenerate images[28]. This is achieved by using a net `x:y:x` with $y < x$, inputs and targets identical. Then, the state of the hidden layer can be regarded as a compressed representation of the image. This approach has clear implications for the texture recognition problem though we do not follow it in this thesis — the reasons for this are discussed below in the context of preprocessing, and touched on in Chapters 3 and 4. Other applications of neural nets include backgammon playing[135], time series prediction[90] (with applications in the physical sciences, such as fluid dynamics, and in economics and financial circles) and hand written ZIP (post) code recognition[92].

Neural nets have been used for many other purposes. Particularly relevant to the current discussion are applications in the petroleum industry, of which

there have been many studies. Neural network models have been used to identify formations from well log data[128, 2] and performed well compared with other techniques for such a purpose[145]. More generally, they have been used to compress seismic images[35] (compare [28]), to detect spikes in data[104] and recognise fractures[124]. On a wider scale of operations, neural networks have been used to predict reservoir permeability[108]. They have been used for the problem of texture classification in remotely sensed data[60] but not employing the cooccurrence based measures used here.

2.1.5 Comparison with Traditional Statistics

There is a close relationship between neural networks and traditional statistical classifiers. This is discussed in [121]. A more extensive discussion is found in [8], which compares neural networks with the discriminant functions used by statisticians. The perceptron architecture described above, for instance, bears comparison with Fisher's linear discriminant function[45] which assigns a vector $\mathbf{x}$ to one of two classes depending on the sign of

$$\mathbf{y}^T.\mathbf{x} + y_0$$

where $\mathbf{y}$ is also a vector. Comparisons are also made between more complicated neural network architectures and other types of statistical classifier. An important difference is that statistical classifiers are constructed to suit particular problems, requiring analysis of those problems. Neural networks do not need to be specified so tightly — though as we saw above, knowledge of the problem may be required if the network is to be efficient.

2.2 Texture

As well as neural network techniques, there exist equivalent classical statistical techniques for classification. For examples of the use of these, see [120, 107] the first of which deals with object recognition and the second with textures. (The examples often used in the literature are those given in Brodatz's book[15].) We now turn to the task of texture recognition, since a dominant property of the images which concern us is texture.

2.2.1 The Nature of Texture

The concept of texture is an everyday one, and its meaning is intuitive; it is not easy to define precisely. Collins English Dictionary[65] defines texture as

> The representation of the nature of a surface...

We will therefore adopt a threefold classification of elements used in human perception which is given by Haralick[67]. This scheme distinguishes between **spectral features**, which describe average tonal variations, **textural features**, containing information about the spatial distribution of tonal variations within a band and **contextual information** which is derived from the area surrounding that which is being interpreted. The thesis will focus on textural information, so the images treated are assumed to be homogeneous; their textures are regarded as constant. The measures used to quantify texture are based on such an assumption as they are defined over areas of image. If contextual information were used, we would distinguish between areas in the image with different (local) textural properties. The distinction is somewhat arbitrary and depends on the length scale being used — the concepts of texture and tone are complementary[66] since the tone (overall brightness/darkness) dominates when the variation in a small region is not great while texture becomes important as the amount of this variation

increases. This means that the two concepts can never be wholly separated.

2.2.2 Segmentation and Classification

Interpretation of an image, whether from textural cues or by other means, can be seen as a process of **segmentation** and **classification.** The former involves finding regions with similar properties, eg "smooth" or "rough" areas in an aerial photograph ("finding objects"). Classification is the process whereby these regions are assigned to known types of feature which can be considered homogeneous on the scale of the image — lakes, rivers, grassland, forest, city and so forth ("identifying objects"). The work in this thesis is concerned wholly with the classification problem since the image data is presegmented; we are given separate photographs whose classification is known.

2.2.3 Statistical and Structural Approaches

There would seem to be two ways to describe a texture; one could specify some sort of micropattern (usually described as a **primitive**) which is then distributed according to placement rules, or some sort of statistics[2] might be used. These concepts merge if the primitive (the simplest form of which is a pixel) is distributed probabilistically rather than being placed according to deterministic rules. The introduction to [116] discusses texture in terms of repeated substructures and also in terms of stochastic patterns. To illustrate this, we give examples of the two paradigms for texture description, one generated stochastically, the other by means of placing a micropattern. Analysis has often assumed the latter, because this is easier to treat. However, many real world textures are better described by the former approach and attempts have been made to generate textures by

[2]For example mean, variance, variogram, correlation length, or a distribution function for the greyscale values

stochastic means[127, 123].

We present two contrasting examples of textures. Figure 2.4 was generated by the method of [127][3] while Figure 2.5 was generated by random placements of a simple subpattern.

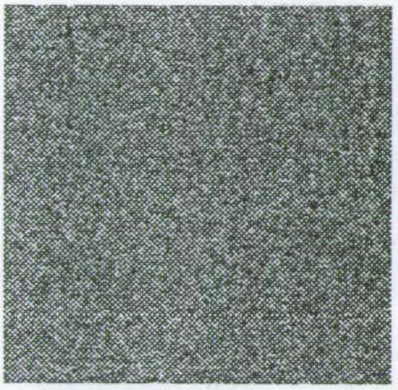

Figure 2.4. Stochastic Texture. This image was generated by the method of Smith and Freeze[127] for creating correlated random fields.

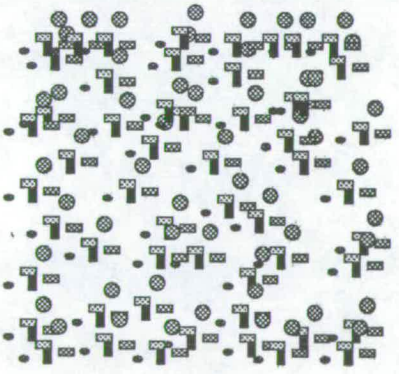

Figure 2.5. Texture Generated by Subpattern. This image was produced by smie randomly placing a sub unit.

[3]I am grateful to Gillian Pickup of the Department of Petroleum Engineering at Heriot-Watt University for making her software available to me

The concepts of tone, of texture, of primitive and of statistical and structural descriptions are closely related. Except in the simplest cases (such as a blank image) one may usefully describe texture by the number and types of primitives, and by their spatial orientation and layout[66]. Most work concentrates on one or another of these aspects. We will take the primitives as given (greyscale levels) and consider their spatial relations. Before exploring the correct framework for this, since texture discrimination can certainly be performed by biological systems (humans!) we examine their capabilities, as these will help us motivate our approach.

2.2.4 Biological Motivation.

What do we mean by "different" textures? The authors of [116] describe texture measurement as a "nebulous" subject — most definitions have a local pattern and nearly periodic repeat[113, 71]. However, this is limited to deterministic textures — natural ones have a stochastic structure [67, 68]. In the absence of any all embracing model, as for instance we have with colour[4] as a starting point we can say: Textures differ if **people** can distinguish them. This definition focusses attention on human texture processing capability. We assume that this capability evolved to meet some need in our past — most likely concerned with avoiding predators or finding food. Thus, we will be able to discriminate certain textures, and not others.

It will therefore be of some use to consider briefly work which has been done to investigate and clarify human texture discrimination abilities. We hope that this will give some indication of how such discrimination may be imitated artificially.

[4]Where we can relate a perceived impression ("red") to a physical quantity ("frequency of light")

By "discriminate" we are following Julesz[78] and others and referring to **preattentive** texture discrimination, that is "instant", at a glance, discrimination. It is sometimes possible, by means of poring over examples of textures, for a human to work out that they are different, but this is another matter altogether since it involves conscious effort. (It is not clear that this difference is important for artificial methods of texture discrimination.)

The distinction noted above between statistical and structural approaches to texture is apparent through many years of research into human capabilities. A good summary of this research may be found in [81].

Julesz[78] used Markov processes to create textures with identical second order statistics and showed that they were usually indiscriminable, but that local properties (features other than the global statistics) were also important. In [80] Julesz conjectures that

> textures that differ in their first and second order statistics can be discriminated from each other, those that differ in their third or higher statistics usually cannot.

([78] supports this view, but used textures subject to various mathematical constraints; [80] used various additional methods to generate textures, getting round both these constraints and the problem of only using 1D Markov processes.) Some weak counterexamples to the conjecture above are found, but on the whole it appears that second order statistics are important for preattentive texture discrimination by humans. This conclusion is supported by [116], which gives an algorithm for generating fields with correlated random numbers. See [115] for generation, [142] for definition of a stochastic field

> An array or numbers randomly distributed and governed by some joint probability density.

Based on these textures, the conclusion was then that the Julesz conjecture holds for some spatially correlated patterns, but mean, variance and higher moments are not sufficient as a description for texture.

In [86], however, Julesz exhibits a pair of textures that have identical third order statistics, and thus identical second order statistics, and the same power spectrum. Nevertheless, clear discrimination is possible, based on a difference in "granularity". This is evidence that some form of local feature is being detected — such elementary features may be termed **textons.** [83] and [84] show that elementary "quasi collinearity", "corner" and "closure" features can be used to discriminate textures. [86] also shows that "granularity" does not depend on second order statistics, as had previously been assumed.

It is now known that human texture discrimination is not wholly describable by statistics. [82] makes it clear that texture discrimination is based on differences in **texton density** (a first order property). The original Julesz conjecture, was shown to be wrong by the creation of counterexamples; it was modified to

> The preattentive textural system cannot globally compute third or higher order statistics.

[82] goes further:

> The preattentive textural system globally can evaluate only the first order statistics of textons.

Julesz identifies the following textons; line segments, elongated blobs and line terminators. (This neglects colour.)

2.2.5 Artificial Methods.

From the above research, it would seem that a biologically inspired texture recognition system would detect textons. The role of second order statistics, though

important, is not central to this task. However, some workers have obtained good results from methods based on second order statistics. Their work is considered below, after alternatives have been discussed.

Many methods which one might use to extract useful information from a texture field are described in [13]. There are many different approaches. From a structural perspective, one might concentrate on the identification of micropatterns, and then find their distributions. This might be achieved either by searching for known groups of pixels, perhaps using a mask, or more subtly by the use of filters or other mathematical devices. Fourier transforms could detect particular spatial frequencies, for instance; Turner[138] suggests that filters based on Gabor functions[49] may function as detectors of textons. [46] implements a model based on these filters, achieving texture discrimination that is good but below the standard of human human performance. See also [137]. The Gabor functions have a close affinity with wavelets[21], whose properties may prove useful in problems such as time series analysis. Alternatively, Pentland[112] uses a fractal description for various tasks, including image segmentation and texture generation, and notes[112, p. 978] that

> ...the fractal dimension of the image data can be measured by using either cooccurrence statistics ...or by means of the Fourier power spectrum ...

The cooccurence matrix based measures are used in this thesis (§2.3) so it is interesting to see them related to other techniques, involving fractals and the power spectrum. Finally, Faugeras and Pratt[41] use a stochastic field model of texture and extract features derived from the autocorrelation.

Having briefly reviewed some alternative techniques, we now proceed to discuss our chosen method for texture characterisation.

2.3 Spatial Grey Level Dependence Method

We have seen above that second order statistics, of which the cooccurrence matrix based measures (or SGLDM) are an example, are not of ultimate importance in human texture discrimination. They are, though, valuable in characterising which textures are discriminable. This fact has been used by various workers as motivation for using the SGLDM to identify textures. Indeed, this is the purpose for which Haralick[67] introduced the method.

Conners and his collaborators have successfully used these measures for the segmentation of an aerial photograph[24]. In [25] they claim that the SGLDM

> ...is the most powerful statistical texture analysis algorithm.

They also claim[24, 23] that the method can successfully discriminate texture pairs that were counterexamples to the Julesz conjecture[5] and conclude that

> ...there is no known example of a visually distinct texture pair which cannot be discriminated by the cooccurrence matrices.

This statement is taken up word for word by Booth[13]. These are strong claims for the success of the method. There have also been studies comparing the SGLDM favourably to other techniques [101, 33].

The method is also used by Oja et al[107] in a study of classification of the Brodatz textures[15], by McCauley and collaborators [141, 97], by Shearer and Holmes[126] in identifying plant canopies, and by Han and Hayes[64] for soil cover. Carstensen[20, 19] also uses the method to discriminate Brodatz textures, employing a binary decision tree. Other successful applications of the SGLDM are to be found in [13]. There are enough of them to make the method credible. This approach is further developed in [22], where additional measures are defined in

[5] Pairs of textures with the same second order statistics that were visually distinguishable

order to retain more information from the original image in the texture measures. Work on efficient calculation of the measures is described in [111].

2.3.1 Notation

Let an image I consist of $L_x \times L_y$ pixels or sites, so that $x \in [0, L_x - 1]$, $y \in [0, L_y - 1]$. There are N_G greyscale levels so that $S_I(x,y) \in [0, N_G - 1]$ represents the states of the pixels. Define first an average of some function $f_I(x,y)$ over such an image.

$$f_I = \overline{f_I(x,y)} = \frac{1}{L_x L_y} \sum_{x=0}^{L_x-1} \sum_{y=0}^{L_y-1} f_I(x,y) \tag{2.9}$$

It is useful to consider also an ensemble average over a distribution $D(I)$ of images I.

$$F(x,y) = \langle f_I(x,y) \rangle = \frac{1}{Z} \sum_I f_I(x,y) D(I) \tag{2.10}$$

where $D(I)$ is a distribution of images and Z is the normalisation.

$$Z = \sum_I D(I) \tag{2.11}$$

2.3.2 The Cooccurrence Matrix.

The matrix element $P_I(x',y')_{ab}$ represents the probability that if a random site (x,y) in the image is chosen, it will have greyscale level a and the pixel at $(x + x', y + y')$ will have greyscale level b. Using the notation introduced above, one can write

$$P_I(x',y')_{ab} = \overline{\delta(S_I(x,y),a)\delta(S_I(x+x',y+y'),b)} \tag{2.12}$$

where the image I is represented by the integer valued function $S_I(x,y)$, and the Kronecker delta is

$$\delta(a,b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{if } a \neq b \end{cases} \tag{2.13}$$

The cooccurrence matrix can also be defined on for 1D "images"; and from (2.10) its expectation value can be determined, if we have a distribution of images; for instance, if the images are generated by a Hamiltonian process, such as the Ising model (Chapter 5).

The matrix $P_I(x', y')$ is not symmetric in general. It is symmetric for textures generated by most models in equilibruim statistical mechanics — see Chapter 5. One may define a symmetric cooccurrence matrix[67] by averaging $P_I(x', y')$ and $P_I(-x', -y')$.

2.3.3 Nth Order Statistics

Before proceeding, we clarify the distinction between various orders of statistics. Given nth order statistics, lower order ones can be found. For example, we have the matrix $\mathbf{P}(x', y')$ (henceforth denoted $\mathbf{P}$ with elements P_{ab}) which contains second order information, that is, a joint distribution. We wish to derive the first order distribution P_a which represents the relative frequency of pixel value a for image I.

P_{ab} was defined above (2.12). One can similarly put

$$P_a = \overline{\delta(S(x, y), a)}$$

which is the probability that a given pixel has value a. It is then easy to show, by substituting for P_{ab}, that

$$P_a = \frac{1}{2}\sum_b (P_{ab} + P_{ba}) = \sum_b P_{ab} = \sum_b P_{ba}$$

And since this is true for all (x', y')

$$\sum_b P(x', y')_{ab} = \sum_b P(x'', y'')_{ab}$$

since the first order statistics (pixel frequency) must be the same, whatever (x', y') are used, if the images are identical.

2.3.4 Functions of the Cooccurrence Matrix Elements

Let M be a function of the cooccurrence matrix elements, that is, a texture measure. M is thus a function of the image and of the parameters x' and y' used to calculate the cooccurrence matrix.

$$\begin{aligned} M(x', y') &= \sum_{a,b} f(a,b) P_{ab} \\ &= \sum_{a,b} f(a,b) \overline{\delta(S_I(x', y'), a) \delta(S_I(x + x', y + y'), b)} \\ &= \frac{1}{L^2} \sum_{x,y} \sum_{a,b} f(a,b) \delta(S_I(x', y'), b) \delta(S_I(x + x', y + y'), b) \\ &= \overline{f(S_I(x,y), S_I(x + x', y + y'))} \end{aligned} \tag{2.14}$$

This result makes it clear that the texture measures are functions of the image and of the displacement (x', y') within it.

2.3.5 Texture Measures in Common Use.

The five measures defined below (energy E, entropy H, correlation C, local homogeneity L and inertia I) have been used extensively. They come from the original Haralick paper [67]. (We will write P_{ab} for $P(x', y')_{ab}$.) These names are frequently encountered in the texture recognition literature, though (confusingly) others are also used. They do not all relate directly to the normal meanings of

the terms.

Energy

$$E(x',y') = \sum_{ab}(P_{ab})^2 \tag{2.15}$$

The energy measures the "sameness" of the image (or lack of it) — the image homogeneity[13]. Clearly, the maximum value that any element P_{ab} can have is 1, when the image is blank. In any other case, the elements of P will all be less than 1 and so will the sum of their squares. Thus, E will be less than 1. For a blank image then (or any image when $(x',y') = (0,0)$), $E = 1$; otherwise $E < 1$. There is a relation with the variance of P, σ_P.

$$E = 1 + N_G^2\sigma_P^2$$

Entropy

$$H(x',y') = -\sum_{ab} P_{ab}\log(P_{ab}) \tag{2.16}$$

Correlation

$$C(x',y') = \frac{1}{\sigma_x\sigma_y}\sum_{ab}(a-\mu_x)(b-\mu_y)P_{ab} \tag{2.17}$$

This measure is nothing other than a cumulant. Booth[13] describes it as

> ...a measure of the degree to which rows (or columns) resemble one another.

Local Homogeneity

$$L(x',y') = \sum_{ab}\frac{1}{1+(a-b)^2}P_{ab} \tag{2.18}$$

Because of the factor $\frac{1}{1+(a-b)^2}$, this measure gives a greater weight to the on diagonal elements of the cooccurrence matrix. These elements represent self correlations between pixel values, so the local homogeneity feature would appear to detect homogeneity, as the name implies. From (2.14) any significant correlation between $S(x,y)$ and $S(x+x',y+y')$ will increase the magnitude of the denominator and thus reduce the value of the measure. So a plot of $L(x',y')$ should have minima where there is some correlation.

Inertia

$$I(x',y') = \sum_{ab}(a-b)^2 P_{ab} \tag{2.19}$$

The inertia measure has a relation to the periodicity. This should be clear from considerations similar to those above. Conners[22] discusses this. There is a relation with the autocorrelation function[14]

$$I(x',y') = \sum_{ab}(a-b)^2 P_{ab} \propto \gamma(x',y')$$

which means (since the autocorrelation function is the Fourier transform of the power spectrum) that there is a relation between inertia and the power spectrum and also between inertia and the "variogram", which is often used in geostatistics. [66] comments that directional textures should have peaks with ϕ while blobby ones should have peaks with $\mathbf{r}$, where $(\mathbf{r},\phi)$ are polar coordinates. Booth[13] describes inertia as "a measure of the spread of matrix values." [79, 24]. This description would seem to fit the energy measure better. It is easy to show that for $x' = y' = 0$, inertia is 0.

In (2.15) — (2.19)

$$\mu_x(x',y') = \sum_a a \sum_b P_{ab} \tag{2.20}$$

$$\sigma_x^2(x',y') = \sum_a (a-\mu_x)^2 \sum_b P_{ab} \tag{2.21}$$

with similar definitions for μ_y and σ_y^2.

Conners et al[25, 22] introduce two further measures, **cluster shade** A_3 and **cluster prominence** A_4. They claim that adding these to the feature set consisting of the five measures defined above retains more information. (Note also references in [25, 22] which give properties supposedly captured by these measures.)

Cluster Shade

$$A_3(x',y') = \sum_{ab} (a+b-\mu_x-\mu_y)^3 P_{ab} \tag{2.22}$$

Cluster Prominence

$$A_4(x',y') = \sum_{ab} (a+b-\mu_x-\mu_y)^4 P_{ab} \tag{2.23}$$

We do not use these features in our training.

2.3.6 Form of P_{ab} for Special Cases.

It is possible to calculate the form of the cooccurrence matrix (and so the values of the texture measures) in some simple cases. This is worth doing as bounds are thereby set on the behaviour of the measures. In Chapter 5 exact values are calculated for some simple models of images, and these can be compared with the cases discussed below.

Long Range Order

We consider the behaviour of the cooccurrence matrix, and the texture measures, as $|x'|, |y'| \to \infty$.

Correlation If an image is statistically translationally invariant, as we assume our rocks to be, then $\langle S(x,y)\rangle = \langle S(x+x',y+y')\rangle$. From (2.17)

$$C(x',y') = \frac{1}{\sigma_x\sigma_y}\left(\sum_{a,b} abP_{ab} - \mu_x\mu_y\right)$$

For any function $f(a,b)$, (2.14) states that

$$\sum_{a,b} f(a,b)P_{ab} = \frac{1}{L^2}\sum_{x,y} f(S(x,y), S(x+x',y+y')) \tag{2.24}$$

which defines a mean value

$$F(x',y') = \overline{f(S(x,y), S(x+x',y+y'))} \tag{2.25}$$

over the $L \times L$ image ($S(x,y)$ the image function itself). Thus,

$$\sum_{ab} abP_{ab} = \overline{S(x,y)S(x+x',y+y')} \tag{2.26}$$

Then as x',y' increase,

$$C(x',y') = \overline{S(x,y)S(x+x',y+y')} - \mu_x\mu_y = \gamma(x',y') - \mu_x\mu_y \tag{2.27}$$

(where $\gamma(x',y')$ is the autocorrelation function[14]) so C goes to 0.

Inertia As before, the term in abP_{ab} is equal to $2\overline{S(x,y)S(x+x',y+y')}$ and tends to $2\overline{S(x,y)}^2$ as x',y' increase and $S(x,y) \rightarrow S(x+x',y+y')$. By Equation (2.24) the term $\sum_{ab} a^2P_{ab}$ gives $\overline{S(x,y)^2}$; similarly the term in a^2 gives $\overline{p(x+x',y+y')^2}$. As x' and y' grow, these terms become identical so that for large (x',y')

$$I(x',y') = 2(\overline{S(x,y)^2} - \overline{S(x,y)}^2) \tag{2.28}$$

which tends to a finite limit.

This limiting case $|x'|, |y'| \rightarrow \infty$ is equivalent to the case of a totally uncorrelated image, ie a random image. The correlation measure for this case is shown in Figure 2.6 — clearly it falls off rapidly with distance.

Random Image

In this case, we assume that all pixel values are equally likely at any given site and that there is no correlation between them. Thus

$$P(x',y')_{ab} = \frac{1}{N_G^2} \tag{2.29}$$

The results obtained upon this basis will agree with the "large distance" values obtained above if the latter also have all pixels equally likely and finite range correlations. In fact, the former condition is unlikely to be satisfied by real images. Assuming that (2.29) is valid for all (x', y') gives

$$E(x',y') = \frac{1}{N_G^2} \tag{2.30}$$

$$H(x',y') = -2log(N_G) \tag{2.31}$$

$$C(x',y') = 0 \tag{2.32}$$

$$L(x',y') = \frac{1}{N_G^2}\left(N_G + 2\sum_{k=0}^{N_G-1}\frac{N_G-k}{1+k^2}\right) \tag{2.33}$$

$$I(x',y') = \frac{1}{6}\left(N_G^2 - 1\right) \tag{2.34}$$

$$\mu_x = \mu_y = \frac{1}{2}(N_G - 1) \tag{2.35}$$

$$\sigma_x^2 = \sigma_y^2 = 0 \tag{2.36}$$

We assume that the pixel values are $0, 1, 2, \ldots, N_G - 1$. Different results are obtained for some measures if, for instance, we use pixel values $-\frac{N_G}{2}, -\frac{N_G}{2} + 1, \ldots, -1, +1, \ldots \frac{N_G}{2}$ as in the Ising case of Chapter 5 ($N_G = 2$.) Then those measures whose definitions include a and b will change; for instance,

$$I(x',y') = \frac{1}{6}(N_G + 1)(N_G + 2) \tag{2.37}$$

rather than the result given in (2.34.) With this qualification, these results are consistent with the calculated results for the zero field Ising model (5.31) when $|j - i| \to \infty$. [(5.38), (5.39), (5.40), (5.41), (5.42).]

Short Range Limit

In the short range limit, $x' = y' = 0$, we find that the cooccurrence matrix is highly ordered. Pixel values are correlated only with themselves, so that P_{ab} is diagonal

$$P_{ab} = \delta(a, b)P_a$$

and depends only on the first order distribution of pixel values.

Lined or periodic image

Consider periodic images such as that defined by $S(x, y) = A\sin(\alpha x + \beta y)$. Such images will be totally correlated in one direction. Correlation in a direction perpendicular to the first will vary with displacement and this variation will be periodic. The texture measures all reflect this strong periodicity. In principle, we can calculate the measures from

$$P_{ab} = \frac{1}{L_x L_y}\int_0^{L_x}\int_0^{L_y} \delta(S(x, y), a)\delta(S(x + x', y + y'), b)dxdy$$

Figure 2.7 shows the correlation measure plotted against a displacement (x, y) for an original image which was sinusoidal in form, with spatial frequencies μ_x and μ_y The figure is rather spiky from the interpolation, but it is clear that the texture measure is capturing the periodic nature of the image, as one would expect for a correlation function.

2.3.7 Effect of Rescaling the Image

Before calculating texture measures from the images, we need to consider whether (and how) they should be normalised. The images used vary in their overall tone, and we wish to remove this difference since it is not relevant to discrimination.

The average grey level of an image depends both on the lighting conditions in which the photograph was taken, and on the colour of the rock itself. The colour does not matter; similar lithofacies may appear with different colours depending on whether they were formed in oxidising or reducing conditions.

We consider briefly the effect on the texture measures of some simple operations performed on an image. Suppose that the image is scaled

$$S(x,y) \mapsto \alpha S(x,y) + \beta \tag{2.38}$$

Then α controls the degree to which the cooccurrence matrix elements are "squashed". β controls a translation of the values, or a re labelling of the cooccurrence matrix rows and columns; if $\alpha = 0, \beta = 2$ we obtain a new cooccurrence matrix with labels $N_G \ldots 2N_G$ instead of $0 \ldots N_G$. In this last case, those functions of the cooccurrence matrix (such as texture measures) which do not contain pixel values explicitly will not be changed. Those functions which do contain pixel values will be altered. The use of the "equal probability quantisation"[67] is often recommended. (See the discussion in Chapter 4 concerning this.)

Consider a different type of scaling. The original image is "blocked" by a factor b so that squares of side l are averaged. This produces a new $\frac{L}{l} \times \frac{L}{l}$ image from which a matrix P_{ab} can be derived. An understanding of the effects of this procedure requires a renormalisation group approach[56]. However, intuitively one anticipates that it will bring out features on different scales depending on l. Such features are especially important in work aimed at identifying rocks, since these have structure on many different scales.

In the next chapter we give an account of the first training that was done, and establish the need for some form of preprocessing of the images.

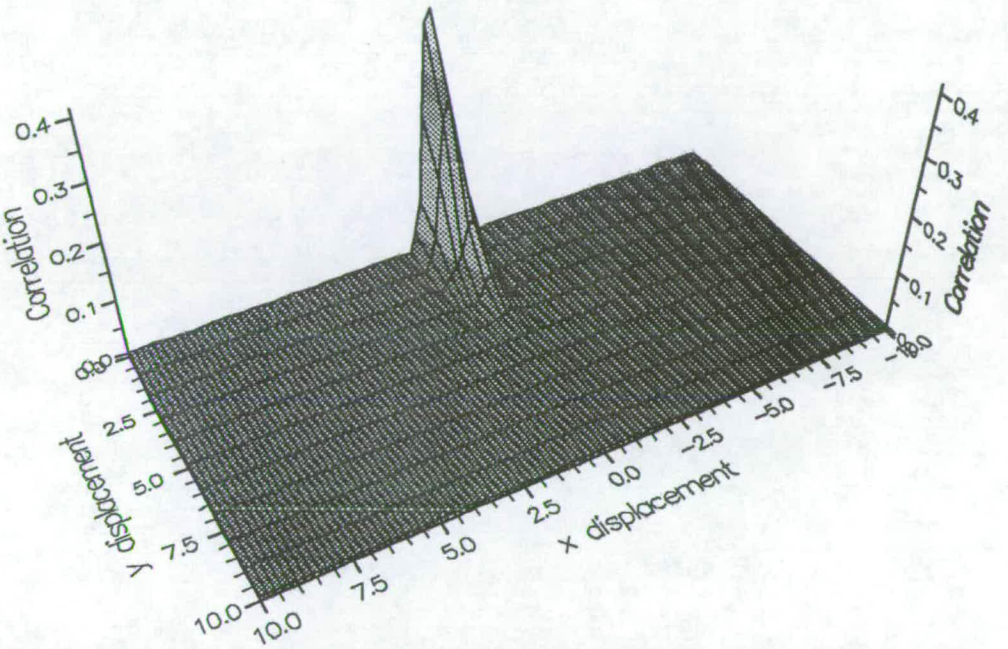

Figure 2.6. Correlation Measure For Random Image. An image was generated with 16 pixel values, equally likely. The cooccurrence matrix was calculated for a variety of (x', y') and the results are shown. The apparent spread in the central peak is due to interpolation to produce the 3D plot.

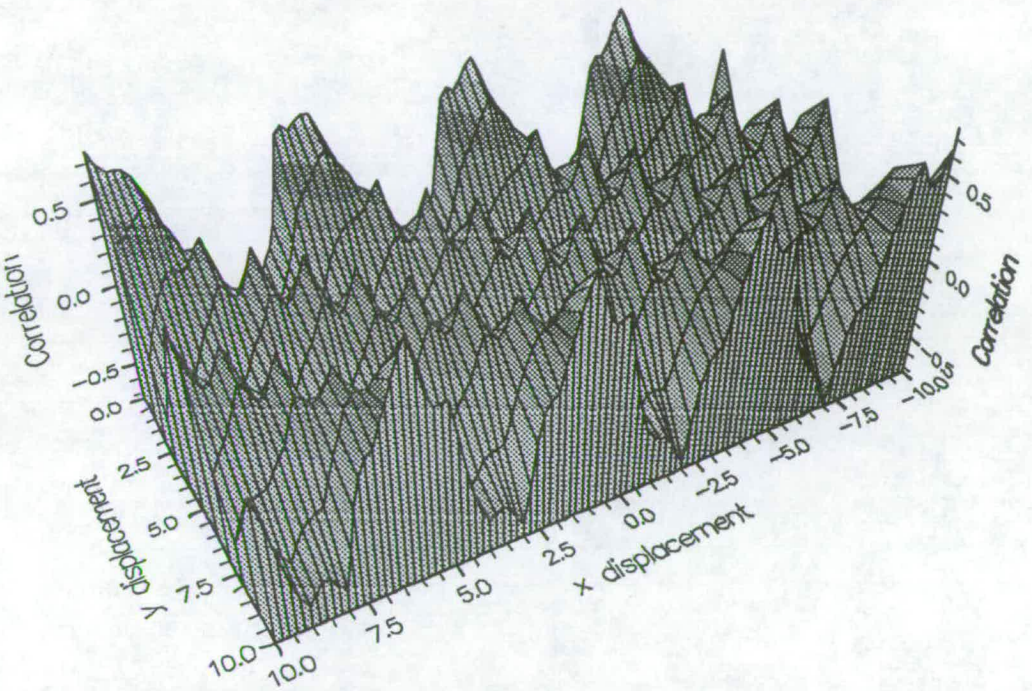

Figure 2.7. Correlation Measure For Sinusoidal Image. An image was generated using a sinusoidal function. Cooccurrence matrices were calculated for various (x', y') and the correlation feature was derived from these. The measure appears to be only approximately sinusoidal; this is because of interpolation to produce the 3D plot.

Chapter 3

Initial Study of Training

This chapter describes the first neural network training that we perform to discriminate between images of two rocks. In this training we use pixel values from photographs of rocks as inputs to one of several different multilayer perceptrons (§3.3). This method has been used successfully before[119, p. 60]. In further training texture measures calculated from the images are used as inputs to networks; it is shown that these measures alone are sufficient for discrimination, even when the network is simplified to a perceptron with only one layer of weights. (§3.4, §3.5). As we see in Chapter 4, a multilayer perceptron is required when there are more than two image classes.

The measures that we use are the the spatial grey level dependence method based measures of Haralick[67]. This method is discussed in Chapter 2, where we see that it has some support from biological research, and that it has been employed successfully in a variety of applications. Chapter 2 also mentions various alternatives that might be used, but which do not seem to have any advantage over the SGLDM.

In performing the preprocessing, it is necessary to choose several parameters which are used in the generation of the cooccurrence matrices. There was not a

sufficient volume of data for a detailed study of all of these parameters, but we present some results on window size and on displacement (§3.6). We find that the measures begin to converge when the window size increases above 256 pixels (1.3×1.3 cm). The dependence of the measures on on direction is not strong for the rock images (unlike some artificially generated images which we discuss below) and it is similar for the different rocks.

To perform network training it is necessary to establish methods for acquiring and preprocessing the data. Before considering the training and the cooccurrence matrix parameters, we review the facilities which were available for these tasks since they had a substantial effect on the project.

3.1 Facilities Available

The rock images which we use are scanned using a Macintosh Quadra machine at Heriot-Watt University. The image data is transferred to Edinburgh University for preprocessing and neural network training. The training is done using several different multilayer perceptron networks, and also a simple perceptron; further training using a multilayer perceptron is discussed in Chapter 4.

During the course of the project many of the facilities available changed (and the skill with which they were used improved). The thesis reflects this. First, the facilities available for data transfer steadily improved. At the start, it was necessary to save the scanned images on floppy discs; then a tape drive became available, and finally an Internet connection was established. These changes allowed progressively greater quantities of data to be used.

Secondly, the software used for determining the texture measures was developed in two stages. We did not discover until after this had been done that a unix command, `pgmtexture`, exists which calculates all 14 measures given in [67], (but

only for four specified angles). Our software was first written for a conventional unix machine[1] and then transferred to a parallel machine[2].

Finally, the neural network simulators used changed. Initially, we expected to use the Rhwydwaith package[118] on the Edinburgh Concurrent Supercomputer, a multiprocessor transputer based machine. This was replaced by the Neurosys simulator[73] on the same machine, but because the network which we eventually built was not large enough to gain advantages of scale from the use of a large parallel machine, the bulk of the training (that described in Chapter 4) was performed on Sun workstations. (PLR training was also done using a conventional serial machine.)

3.2 Pseudo FMS Images

The work in this thesis is motivated (see §1.1) by the need to interpret borehole imagery, and in particular FMS/ FMI — two dimensional "images" that map the resistivity of rock. It is important to work with data that simulate this as closely as possible. It is also important that the images used be unambiguously classified. However, no real FMS images were available. To make up for this lack, high resolution photographs are used of rocks chosen from [93] and [94]. Some slabs of rock were also photographed specially. All of the photographs which we use are shown in Appendix B. They are comparable to real FMS/ FMI images.

To save both time and cost, prints of the actual photographs in [93] and [94] were obtained. This meant some loss of control over quality and over the conditions in which the photographs were taken and processed; it also meant that the area of the images available was rather small in some cases. To compensate

[1] An Edinburgh University Computing Service Sequent
[2] A Thinking Machines Corporation CM-200 in Edinburgh Parallel Computing Centre

for this, the preliminary work reported in this chapter is done using larger pieces of rock which were available to be photographed locally. All of these photographs were printed to 1 : 1 scale.

The three stones used in this initial investigation (all illustrated in Appendix B) are:-

Gatelawbridge (Figure B.1). A red/orange stone with laminations throughout the block on various scales and a fine grained structure. Laminations are at angles varying from the horizontal to $\approx 20°$.

Dunhouse Buff (Figure B.2.) A yellow stone with no laminations, but clearly visible elongated features (rootlets) caused by biological activity in the past; these are ≈ 2 cm long and mainly horizontal, though their angles vary. They are randomly distributed throughout the sample.

Locharbriggs (Figure B.3). A laminated red sandstone, with a coarser structure than Gatelawbridge, and more regular laminations.

3.3 Training Without Preprocessing

This first work is done with Gatelawbridge and Dunhouse Buff stones. The photographs are scanned at 50 ppi (pixels per inch) in monochrome using an Apple Macintosh Quadra machine running the Adobe Photoshop image processing program, and normalised, on the same machine, to the same mean greyscale value (preventing discrimination by means of tone). Regions of 50×20 pixels are cut from the normalised images and saved as raw data (text) on floppy disk. 50 such datasets are taken for each of the two images. They are then transferred to the Edinburgh Concurrent Supercomputer, and used as inputs to a neural network

realised on the neural network package Rhwydwaith.

At this stage, we had not written any software for preprocessing the images. When we did this, we included a capacity to normalise image greyscale values to have a given mean. Doing this means that we can control the way in which the normalisation is performed. On the Apple Quadra, normalisation is done graphically using a mouse pointer and histogram; it is not possible to give a desired value for the mean.

Two approaches to training using the normalised images are explored.

3.3.1 Simple One Layer Network

Rock images of Gatelawbridge and Dunhouse Buff are edited manually into files containing 50 × 20 pixel regions. These files are presented to a one layer, fully connected feedforward network with 1000 inputs, 10 hidden nodes, and 1 output of the type shown in Figure 2.2. Targets are taken as 1 or 0, according to the original rock type.

This net is trained with backpropagation for approximately 20 hours, running on a 17 transputer domain of a Meiko Computing Surface using the Rhwydwaith package[119]. Training is halted after rather more than 160000 update cycles[3] and tests made. The results are not good — the network classifies only 17.5% of the training set correctly.

In such training, we normally expect that performance on the training set should be better than 50% correct (for a binary classification.) However, we were using targets of 1 or 0 with no tolerance — so only those training patterns for which the net produced an output of **exactly** 1 or 0 were classified correctly. With the benefit of hindsight, it would have been better to have allowed some small tolerance of error on the outputs.

[3]In one cycle, every pattern is presented once.

3.3.2 Structured Network

To overcome the poor performance of the first network, we use another, more structured network. This is still a multilayer perceptron, but the network has a grid of input nodes, from which windows of 10×10 inputs are connected to each single hidden node. These windows overlap by 5 nodes on each side, so there are 27 of them in all. The hidden nodes are fully connected to the output node. Figure 3.1 shows the 20×50 set of inputs, (top). A single window is shown, within which all the inputs are connected to one hidden layer node. The output is at the bottom.

This strategy reflects the 2D nature of the data, and the fact that some sort of locality in the texture should be expected, allowing the net to accept regions of the image with similar properties. The first network is fully connected and so it contains the structure of the second within it, rather as a block of stone contains a statue; but training the first network to find the second would take a great deal of time. We are helping the network, using our knowledge of the problem, and so reducing the training that would otherwise be necessary.

Such a technique may be compared with those described in §2.1 for designing a network. By imposing a structure on the network we reduce its complexity. Use of weight sharing or weight decay has the same objective.

From the 100 patterns available, two independent sets of 50 patterns are drawn (sets 1 and 2). One of these is used for training and the other for testing the trained network. This procedure is repeated for a different choice of test and training patterns (sets 3 and 4) drawn from the same 100 patterns.

The second approach is more successful. There are fewer weights — 2727 rather than 10010 — so that the the training time is reduced[4]. Training now

[4]In a fully connected network $1000 : 10 : 1$ there will be $1000 + 10 + 1 = 1011$ nodes and $1000 \times 10 + 10 \times 1 = 10010$ weights. For the non-fully connected network, with 1000 inputs

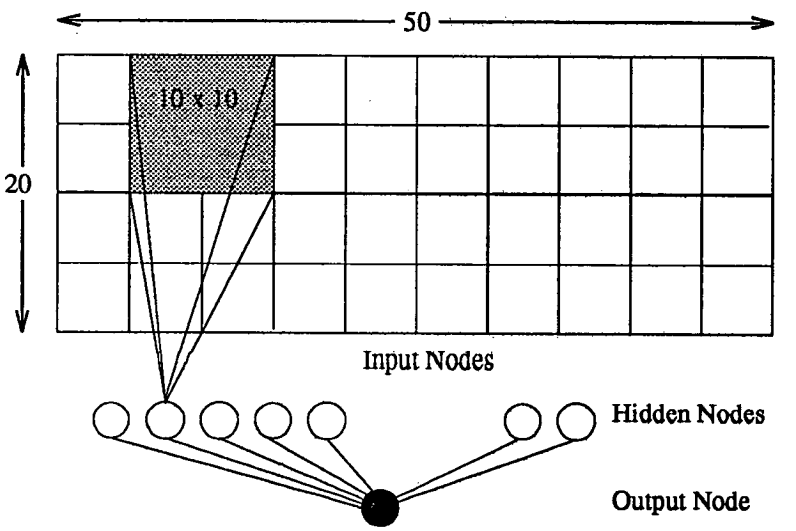

Figure 3.1. Multilayer Perceptron Architecture — Structured Network

takes 200 — 300 update cycles, or approximately ten minutes. Furthermore, the training set is learned with 100% success. Generalisation , however, is poor, 30% — 50% — this is worse than chance. The results are puzzling. We are confident of their reliability but not of their interpretation. We would expect the output to be better than chance.

The results are summarised in Table 3.1, where data sets 1 and 2 were independent of one another, as were sets 3 and 4. The results show that the network is able to learn the training data very easily; but its generalisation is bad.

We did not spend any further time investigating this network. Given the quantity of data available, it was not worthwhile. The poor generalisation may be because the amount of training data was limited. There were only 100 data sets in all. The number of free parameters in the net is roughly equal to the number of patterns which can be stored. Here, this means that ≈ 2727 patterns

going in blocks of 100 to 27 hidden nodes, which are fully connested to 1 output node, there are $1000 + 27 + 1 = 1028$ nodes and $100 \times 27 + 27 \times 1 = 2727$ weights (or free parameters).

Training Set	Cycles to Train	Test Set	% Patterns Correct
1	191	1	100
		2	42
2	230	2	100
		1	37
3	209	3	100
		4	44
4	191	4	100
		3	45

Table 3.1. Learning Results — Structured MLP network using image pixel values as inputs

can be stored *before generalisation occurs.*

Given sufficient training data and a thorough investigation of the space of available networks and parameters, a MLP can be trained to recognise rocks without the need for preprocessing of the images. Richards[119] achieves this for Brodatz textures[15] with a good success rate. This is a robust method, which is not vulnerable to quirks of the preprocessing. We did not have enough data to stimulate generalisation using this approach. Instead, we decided to use much smaller networks, with preprocessing, so that it would be easier to get good generalisation.

3.3.3 Conclusions.

The work described in this section was done early in the project and could be improved upon. (As with many aspects of the project, if it were to be done again now, it would be done differently.) However, some conclusions can be drawn.

Using knowledge of the problem, a network architecture can be designed which is better able to learn the required classifications than a naive approach. (Compare this with §2.1 where we state that one of the advantages of neural networks

over statistical classifiers is that this kind of design work is not needed! It may, though, be useful.) Knowledge of the problem gives other advantages. For instance, it allows use of inherent properties of the data like statistical translational homogeneity, suggesting particular techniques for regularisation. In the sort of problem which we consider here, for example, it might be fruitful to try weight sharing[92, 91] as this would take advantage of the statistical invariance of the rocks.

A particular architecture is useful only if one has sufficient training examples. The number of these required depends on the structure of the network. Any network using pixel vales as inputs will need a very large number of such values in order to capture a significant area of the image. (This will be less true if the image is fairly homogeneous, but in such a case the description "texture" hardly seems appropriate.) This means that even with a minimal architecture, there will be many free parameters in the network. Consequently, many training examples will be required. This will be true even if such techniques as weight sharing are used, though these methods will alleviate the problem slightly.

Unless one has a large quantity of data, something must be done to address this problem. A possible approach is to generate a larger data set by adding noise (from a Gaussian distribution, for example) to the given set of images. Alternatively, one could perform some sort of preprocessing to reduce the size of the input vector.

This preprocessing should respect the symmetries of the data. Krogh, Hertz and Palmer[89, p. 144] discuss two problems — discriminating between odd and even numbers, and finding whether a given number has an odd number of prime factors. Choosing the correct representation of the inputs is crucial; the first problem is easy if the number is represented in binary while the second is easy if the number is represented by its prime factors!

In the next section, we consider the use of preprocessing in the rock texture recognition problem using the SGLDM.

3.4 Training with Preprocessing

The work described above was early, and is of importance because it suggests the need for preprocessing. We now consider what can be achieved with preprocessing.

3.4.1 Multilayer Perceptron

Software was written to calculate values of the cooccurrence matrix based texture measures for nearest neighbour pixels at angles $0°, 45°, 90°$ and $135°$. The values of energy E (2.15), entropy H (2.16), correlation C (2.17), local homogeneity L (2.18), and inertia I (2.19) [67, 13] are used as inputs to a fully connected multilayer perceptron with one hidden layer of three nodes. These measures are calculated using 100×100 pixel windows taken from 1198×1198 pixel regions of the Gatelawbridge (Figure B.1) and Dunhouse Buff (Figure B.2) images, scanned specially. (There is nothing special about regions of this size — we used floppy discs to save the images, and this was the largest size that could be stored on a single disc.) This gives 121 datasets for each image.

Calculating nearest neighbour values of five parameters for each of four angles gives twenty inputs to the network. With three hidden units, the rule of thumb discussed above suggests that up to sixty patterns can be stored. The results of training indicate that the network can learn the dichotomy, and that it generalises well. 20000 update epochs are allowed in every case. (See Table 3.2.)

These results are encouraging. To see if they can be improved upon, we use a

Number of Patterns		% of Patterns Correct	
Test Set	Training Set	Test Set	Training Set
40	200	100.00	96.00
80	160	96.25	96.25
120	120	95.85	95.85
160	80	95.00	95.00
200	40	92.00	92.50
220	20	80.45	90.00
230	10	13.47	20.00

Table 3.2. Learning Results — MLP network using nearest neighbour texture measures as inputs. The first two columns show the number of patterns in the test and training sets. The next two columns show the results. This network should be able to store approximately 60 patterns, so the generalisation performance (test set) should improve when more than 60 patterns are in the training set.

simpler neural network model. The same inputs are used as before but the network is a simple perceptron, as shown in Figure 2.1, trained with the perceptron learning rule(2.1). Again, good results are obtained; these are shown in Table 3.3. The results are achieved with many fewer update cycles than are the multilayer perceptron results. The largest number of cycles required is 136. This is because the perceptron learning rule is simpler and quicker than backpropagation. It seems sensible to take advantage of this. We carried out further studies using the PLR, averaging the results over many different sets of training patterns. These are discussed in §3.5 Before doing this, the software for calculating the texture measures was converted to run on the CM-200. This resulted in a considerable saving of time over the Sequent machine used before. (See Appendix A.)

Number of Patterns		% of Patterns Correct	
Test Set	Training Set	Test Set	Training Set
40	200	100.00	100.00
80	160	100.00	100.00
120	120	97.50	100.00
160	80	96.25	100.00
200	40	96.50	100.00
220	20	98.18	100.00
230	10	87.83	100.00

Table 3.3. Learning Results — perceptron using nearest neighbour texture measures as inputs. The first two columns show the number of patterns in the test and training sets. The next two columns show the results. This network should be able to store approximately 20 patterns, so the generalisation performance (test set) should improve when more than 20 patterns are in the training set.

3.5 Simple Perceptron

Photographs of Dunhouse Buff and Locharbriggs sandstones are used. These photographs are rescanned at 300 ppi and areas of 2048 × 2048 pixels saved to tape. The data are loaded into the Connection Machine DataVault.

The DataVault is a rapid access storage device with high capacity channels linking it to the Connection Machine allowing C* objects to be stored. (C* is a programming language used on the Connection Machine. It is an extension to C, providing facilities for manipulating multidimensional objects which are realised on the many processors in the machine. The DataVault makes it possible to save and reload these objects directly, avoiding the need to treat them serially. See [26] for a brief account of the C* language. [72] is a good general description of the Connection Machine itself.)

The cooccurrence matrices are denoted by $P_{ab}(x', y')$ so there are two obvious parameters which have to be considered — the coordinates defining the spatial displacement. It is useful to parameterise the cooccurrence matrix by Cartesian

coordinates (x', y') rather than plane polars (r, θ) because for small r, $r \sin\theta$ and $r\cos\theta$ are rounded down to zero when performing calculations on the computer.

We can also vary the degree of coarse graining. This is a procedure where the image is divided into cells. In each cell, the average pixel value is calculated and this value replaces the separate values. (See Figure 3.2 for an illustration of this.) We expect, from the central limit theorem, that for images with a finite range of correlation, coarse graining will produce images which have a Gaussian distribution of pixel values. Richards[119, p. 67] performs this operation on Brodatz textures and plots the kurtosis of the coarse grained images against the degree of coarse graining. Kurtosis is defined by

$$K = \left(\frac{1}{N} \sum_{i=1}^{N} \frac{(S_i - \mu)^4}{\sigma^4} \right) - 3 \tag{3.1}$$

where there are N image pixels S_i with a mean value μ and standard deviation σ. Kurtosis is zero for a Normal distribution. Thus, the kurtosis of a coarse grained image gives as indication of how close it is to a Gaussian distribution of pixels. A graph on p. 67 of [119] shows that (for the textures considered) kurtosis does not go to zero until coarse graining by a factor of 8-10 has been done (depending on the texture).

Finally, the window size can be varied. Thus there are three different parameters to be considered.

Figure 3.15 shows the correlation measure plotted against both x' and y'. These figures were made by calculating the measure over the whole image, so that a representative area should be captured. The figures show that there is little asymmetry in the variation of the measure for the two orthogonal directions, so that one may neglect either axis.

Texture measures are derived for various values of x'. The images are normalised to have a mean of 7.5 and the pixel values rescaled to lie in $[0, 15]$. The measures are shown plotted against x', with $y' = 0$, in Figures 3.3, 3.4, 3.5, 3.6,

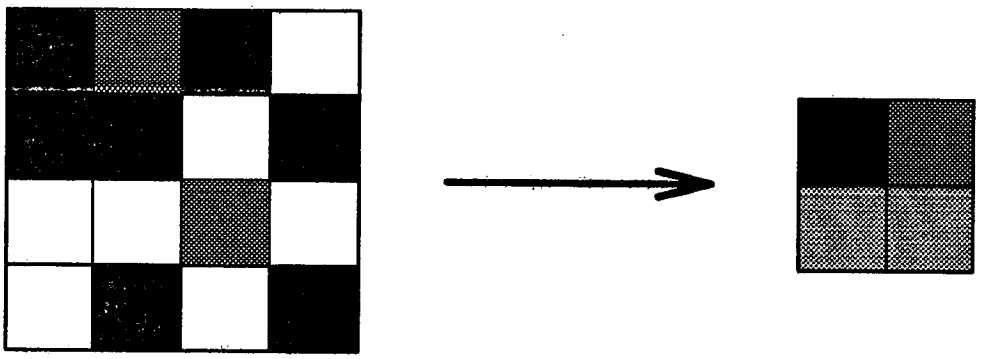

Figure 3.2. Coarse Graining. Groups of 2×2 pixels in the image on the left have been averaged to give the values of single pixels in the image on the right.

and 3.7. The values shown in these graphs are calculated by averaging the results from separate windows of 256×256 pixels across the images. Our principal motive in using this window size was to produce as many data sets as possible, while still keeping the windows as large as they could be; we obtained enough data sets for a standard deviation to be calculated, and also enough data sets to do training. (64 data sets are derived for each image.)

No coarse graining is done on the images. Given the area of image available, this would produce too few training sets.

From Figures 3.3, 3.4, 3.5, 3.6 and 3.7, we can see that the texture measures vary little for values of x' above 8, so the data used for training are derived for $x' = 8, y' = 0$.

The values of the measures at $x' = 0, y' = 0$ depend only on the first order distribution of pixels. (See Chapter 2.) At large distances, the correlations fall off to zero and the measures again depend on the frequency of the different pixels. In order to capture the texture, it is necessary to choose some point between these extremes. Often, the nearest neighbour measures only are used[67] though other authors have studied the effects of varying the distance. In Chapter 4 we

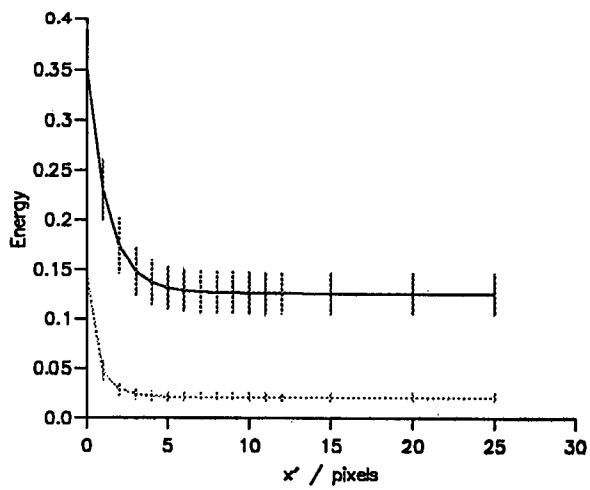

Figure 3.3. Dunhouse Buff and Locharbriggs Energy: $y' = 0$

use measures calculated for a number of values of x' so that the overall shape of the curve is presented.

There are 5 inputs in all, so the network architecture is of the form shown in Figure 2.1 with five inputs and a bias.

Each data set is assigned a target output value — 0 for Dunhouse Buff, 1 for Locharbriggs. From the 128 data sets available, P are chosen randomly and the network trained using them as examples until the error is zero, or 1000 cycles of training have been done. The update stepsize is 1; patterns are presented in random order. Initial weights are random $\in (-1, +1)$. N data sets independent of the training pattern are chosen, and the error in identifying them calculated. The procedure is repeated for different values of P, giving values of $\alpha = \frac{P}{N_W}$ with

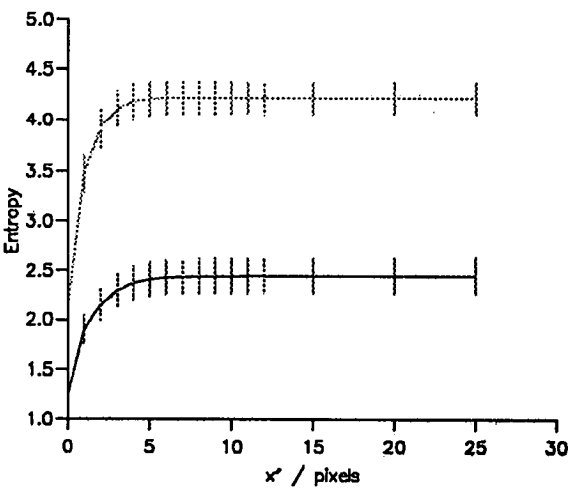

Figure 3.4. Dunhouse Buff and Locharbriggs Entropy: $y' = 0$

N_W the number of weights (6 here, one for the bias).

By carrying out this sequence of steps 1000 times, it is possible to make a graph of error against α, with a reasonable degree of confidence; this graph can be compared to theory. In Figure 3.8 this is done; the different theoretical curves are for different values of ζ, where $(1 - \zeta)$ is a probability.

3.5.1 Results

The form of the graph (Figure 3.8) agrees qualitatively with theory concerning the process of learning a rule. The generalisation error E_G, defined as the fraction of the (unseen) test set that is wrongly classified, begins close to 0.5 for no training,

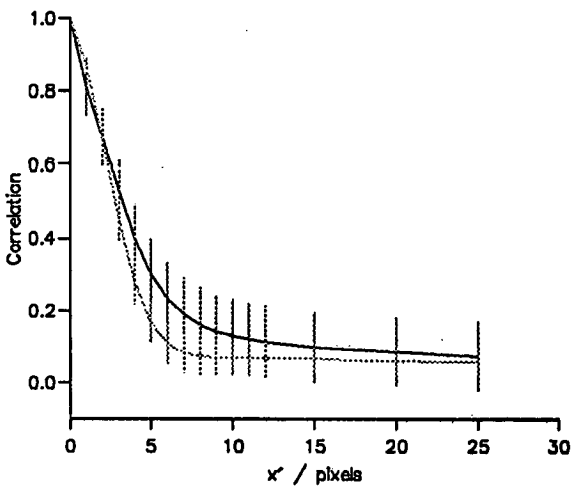

Figure 3.5. Dunhouse Buff and Locharbriggs Correlation: $y' = 0$

as we expect with random initial weights, and falls rapidly as α is reduced through 1.0 to approach some limiting value (here, 0). E_G is always greater than or equal to the training error E_T (within error bounds.) The training error E_T is zero to begin with but increases with the number of training patterns. (This increase only becomes visible for values of α rather larger than 2, off the end of the x-axis in Figure 3.8. For this reason, the training error is not plotted on Figure 3.8.)

These reasons for this behaviour should be obvious. The more patterns drawn from the set representing the underlying "rule" are used for training, the better a representation of this rule will be found. This assumes that such a representation can be found. The clear separation of the curves representing the texture measures (Figures 3.3, 3.4, 3.6 and 3.7) guarantees this. Hence, E_G, which measures

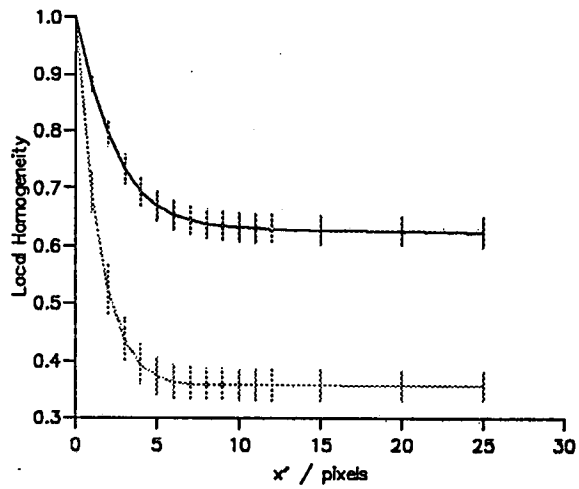

Figure 3.6. Dunhouse Buff and Locharbriggs Local Homogeneity: $y' = 0$

the tendency of the net to incorrectly classify an unseen pattern, will fall as α increases.

On the other hand, the training error E_T describes the tendency of the net to missclassify its training patterns. This should increase with the number of these patterns (since the number of constraints to be satisfied grows) or remain constant, depending whether the training time increases and how learnable the training patterns are.

It is worth comparing these results to theory. Solla [129] gives a bound on the difference between E_T and E_G. We have[129, p. 259]

$$E_G \leq E_T + D(P, h, E_T, \zeta) \tag{3.2}$$

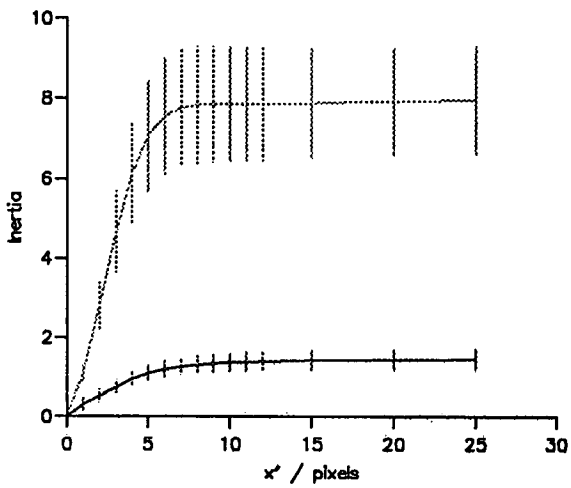

Figure 3.7. Dunhouse Buff and Locharbriggs Inertia: $y' = 0$

with probability greater than $(1 - \zeta)$, P the number of training patterns, h the Vapnik-Chervonenkis dimension of the network. For a simple perceptron, $h = N_W$[89, p. 155]. $N_W = 6$ in this case, with the bias weight, and for E_T close to zero as here, [129] has

$$D(P, h, E_T \approx 0, \zeta) = \frac{2}{P}\left[(\ln\frac{2P}{h} + 1)h - \ln\zeta\right] \tag{3.3}$$

In Figure 3.8, the theoretical curves are calculated for a variety of values of ζ. It is clear that the theoretical and experimental curves agree qualitatively, although the observed performance is better than the theoretical limit.

It may be possible to make this more precise from theoretical considerations. Where the inputs have a large variance (compared to their mean values) it will

be harder for the network to learn the input patterns, if the variance is great enough to imply non-separation of the input parameters for the two blocks. Can the results in such a case be determined analytically? If this is so, then it may be possible to compare different parameter regimes[5] using only the relevant data sets and without performing the training.

Consider Figures 3.3, 3.4, 3.5, 3.6, and 3.7 where all the measures are clearly separated, except for correlation. Though one could use any one of the other measures for training, with a good chance of success, C alone could not be used. This much can be seen from the graphs. If the same conclusion could be reached by means of analysis, then in more complicated cases one might be able to select a good training set other than by trial and error.

In the next section, we consider some more detailed studies on image parameters which were made after the training reported above.

3.6 Studies of SGLDM Parameters

The values of the parameters used in calculating the cooccurrence matrices above were defined in an *ad hoc* way. The window size was chosen to give a reasonable number of data sets, while still (it is to be hoped) being a representative area. The value of displacement was chosen from a comparison of values of measures plotted against x'. Can we make the choice of parameters more systematic?

Here we consider the window size in particular, and look briefly at the effect on the measures of strong correlations in the x' and y' directions.

[5] "Parameters" refers to such factors as the choice of texture measures, the scale of the images, number of greyscale levels retained and so forth. There are other parameters which relate directly to the training process — stepsize, for instance, or length of training.

3.6.1 Window Size

The textural features which we are considering exist on some particular scale. There are variations on other scales than this — the structure of the rocks can consist of various thicknesses of lamina on many scales, and it may exhibit a fractal structure[69] — but there is a representative elementary area which is the smallest portion of the rock distinguishable as being of that type. That is, it captures the essential detail of the image, the features that distinguish it from others and yet is as small as possible for reasons of efficiency. Ideally we will work with samples of images which are this size.

It is to be expected that, as the window size approaches some characteristic size, the measures will converge to representative values.

Procedure

The photograph of Locharbriggs stone is rescanned at a resolution of 500 ppi. This allows us to select four non-overlapping regions of 2048×2048 pixels. Texture measures are derived from these regions using square windows of side 512, 256, 128 and 64. The resolution is chosen to allow the largest window size still to give 16 data sets, enough for a reliable estimate of error to be made for the measures.

It is necessary to use a different piece of the image for each window size. If one used the same piece of image, but the measures were derived with different window sizes, the largest window used (the entire image) would represent the *de facto* limiting values of all the texture measures, to which they would converge. This artefact is known as **arithmetic convergence.**

Results

The results are shown in Figures 3.9, 3.10, 3.11, 3.12 and 3.13. We plot the measure for $x' = 10$, $y' = 0$ against the length of the side of the square window.

The standard deviation is also shown.

In all cases, the measures fluctuate as the window size increases. Since the resolution is 500 ppi, the smallest window size of 64×64 pixels corresponds to a window of side 3.07 mm. This is certainly too small for the characteristics of the rock textures to be visible. However, for energy (Figure 3.9), local homogeneity (Figure 3.12) and inertia (Figure 3.13) the measures are nearly constant when the window side length is doubled from 12.29 mm to 24.58 mm. This tendency is less marked for entropy (Figure 3.10) and correlation (Figure 3.11).

Given the size of the error bars, it would be possible to claim that the values of the measures were constant over the range of window sizes. With a larger quantity of data, these error bars could be reduced. In spite of this, we can conclude that the appropriate scale to use is 256 pixels, that is, 12 mm. or more.

3.6.2 x' and y'

The measures can be understood in terms of the long range behaviour of correlation functions. See [130] for details of this. In Chapter 2 this is shown for the measures used in training; in Chapter 5 the long range behaviour is calculated for a particular example of texture (the Ising model, a simplified model of magnetic spins which is of great interest in statistical physics).

The behaviour of the texture measures, shown in Figures 3.3, 3.4, 3.5, 3.6, and 3.7 is consistent with their limiting behaviour (for example, (2.27) and (2.28) give the limiting form for correlation and inertia and agree with Figures 3.5 and 3.7). This same behaviour is found in the larger set of rock samples (see Chapter 4).

We need to choose values of both these parameters that are "typical" for the images. This may depend on window size. Figures 3.14 and 3.15 shows the correlation feature plotted against varying x' and y'. It is clear that this

feature is, to a good approximation, rotationally symmetric in both cases, though the correlation peak is, wider for Dunhouse Buff than for Locharbriggs. (The measures were derived for the whole images — that is, the windows were 2048 × 2048 pixels — so directional effects were lost).

The two rocks have similar profiles, as might be expected from the earlier graphs where only x' was varied. We had anticipated that Locharbriggs would show a greater correlation in the x' direction because of its laminations, but this was not so, possibly also due to the window size. That such correlation effects will appear is shown by the following.

We generate 10 128 × 128 pixel samples of synthetic textures using the correlated random field method of Smith and Freeze[127] which was employed to create Figure 2.4 in Chapter 2. One of these examples is Figure B.4 in Appendix B. This texture is highly correlated in the x' direction and anticorrelated in the y' direction. Figure 3.17 shows the correlation measure for this texture plotted against x' and y', the other coordinate being held fixed at 0. The effect of the spatially inhomogeneous correlation is clear. In Figure 3.16 the energy measure is plotted for the same texture. In this figure, the inhomogeneity does not appear.

3.7 Conclusions

The work reported in this chapter has demonstrated that, with appropriate preprocessing, simple neural network models can discriminate rock textures. For the calculation of these measures it is necessary to define various parameters, the values of which can be fixed from the images themselves, given a large enough area.

In the next chapter, we use a multilayer perceptron to resolve the more complicated case of discriminating more than two images.

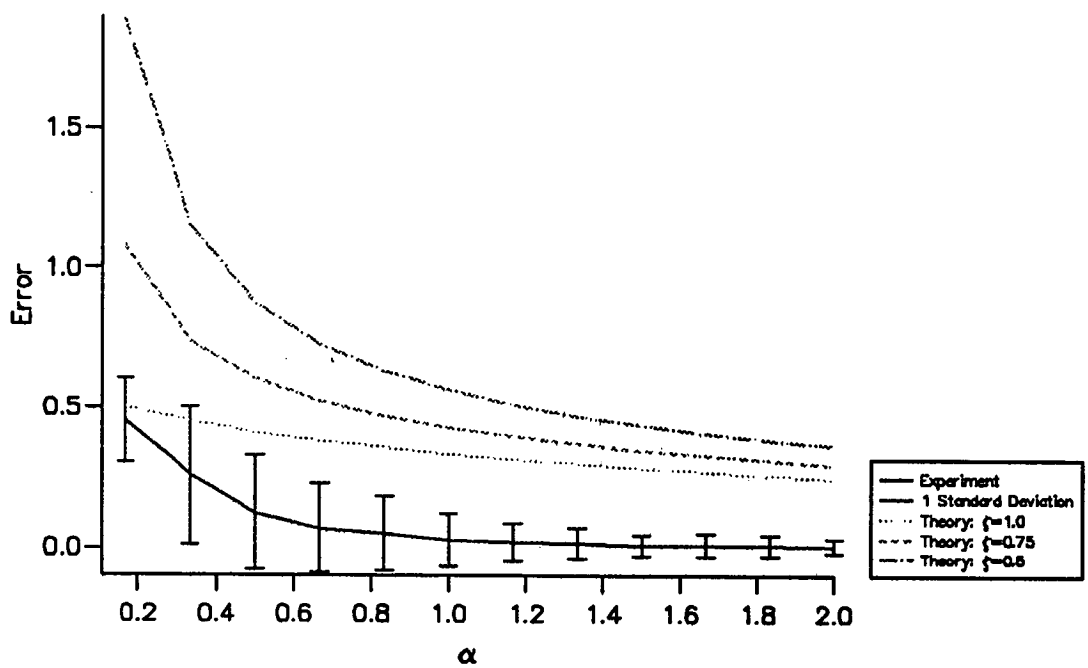

Figure 3.8. Results of PLR training using texture measures as inputs. As α increases, more training patterns are used. Also shown are some theoretical curves relating expected success rate to number of training patterns, derived from (3.3) and using various different values of ζ (where $(1 - \zeta)$ is a probability).

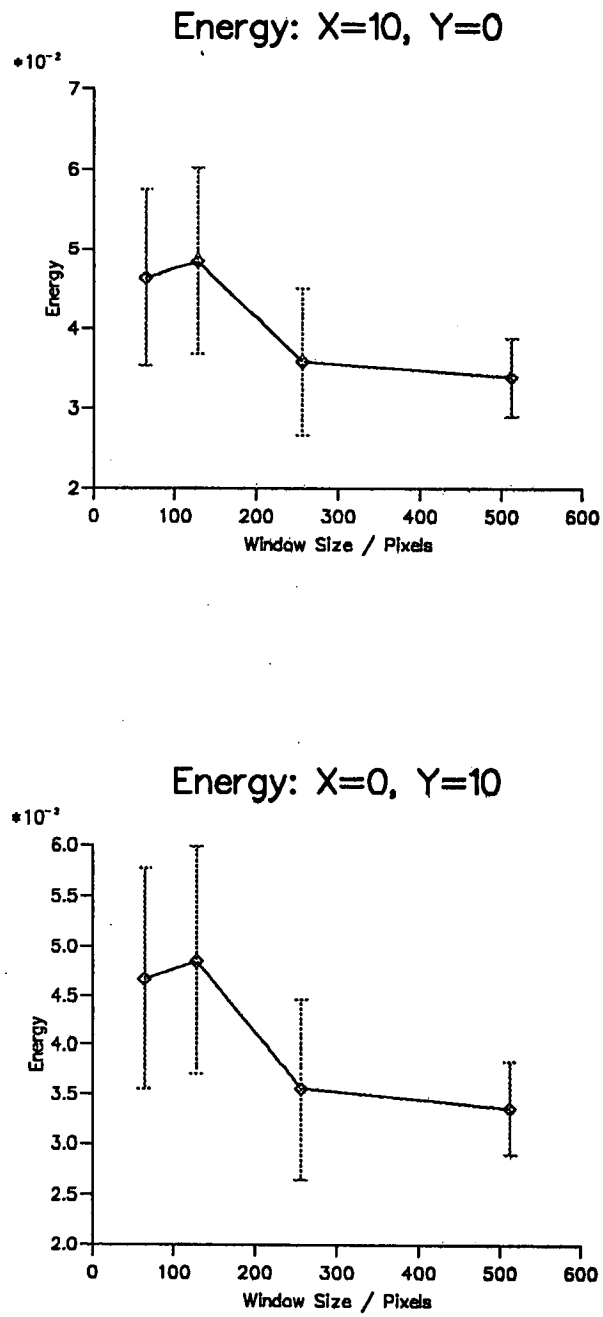

Figure 3.9. Energy Measure: Locharbriggs Stone. The energy is shown for the two specified displacements, calculated for various window sizes. 500 pixels = 1 inch (2.54 cm).

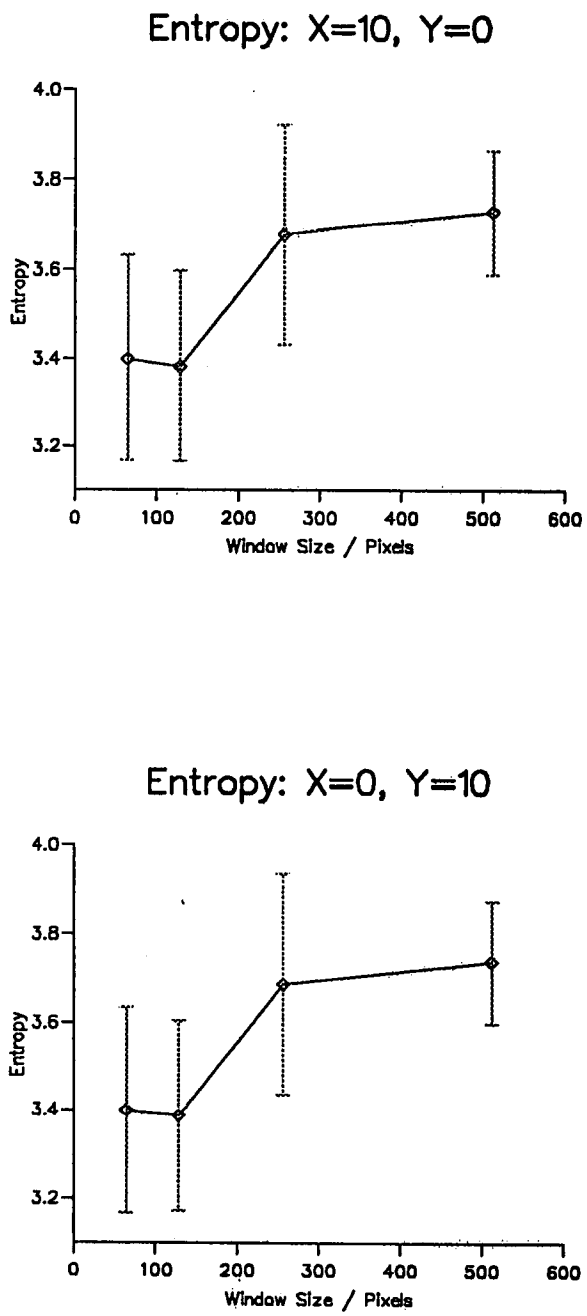

Figure 3.10. Entropy Measure: Locharbriggs Stone. The entropy is shown for the two specified displacements, calculated for various window sizes. 500 pixels = 1 inch (2.54 cm).

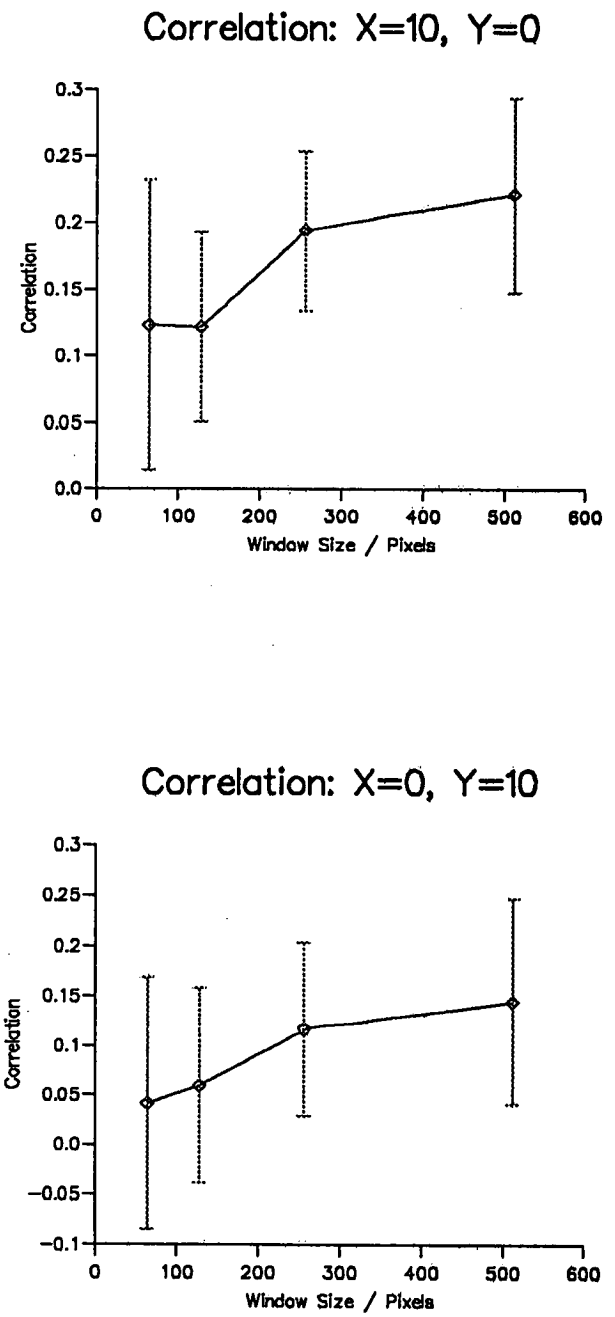

Figure 3.11. Correlation Measure: Locharbriggs Stone. The correlation is shown for the two specified displacements, calculated for various window sizes. 500 pixels = 1 inch (2.54 cm).

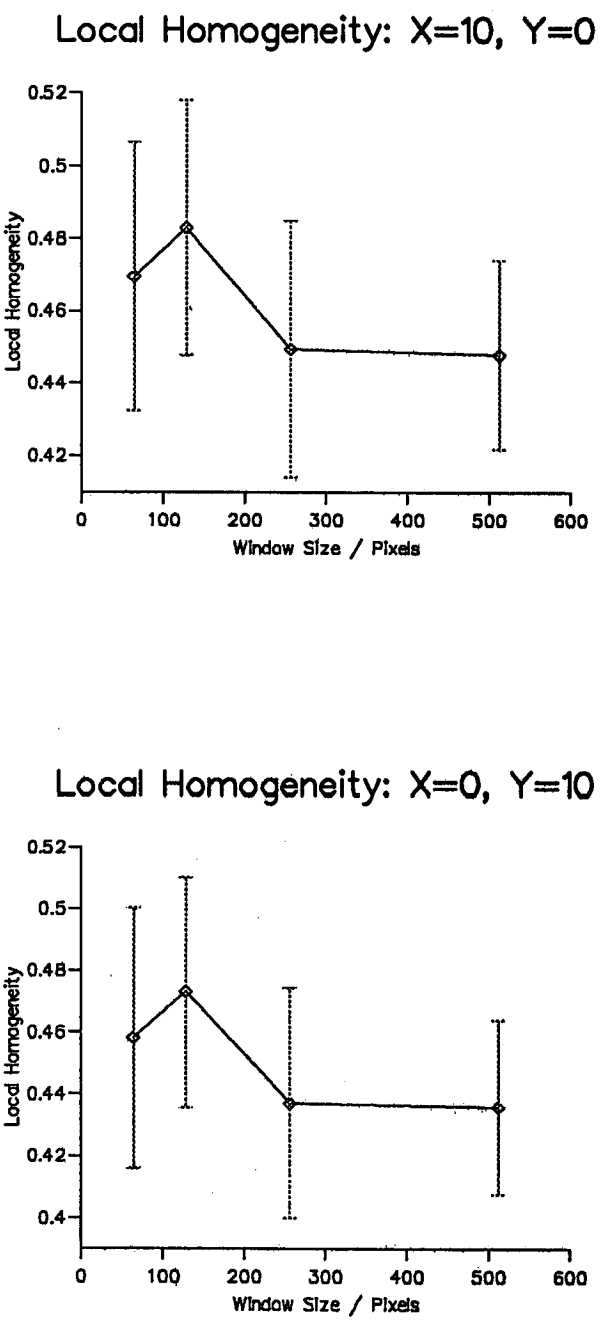

Figure 3.12. Local Homogeneity: Locharbriggs Stone. The local homogeneity is shown for the two specified displacements, calculated for various window sizes. 500 pixels = 1 inch (2.54 cm).

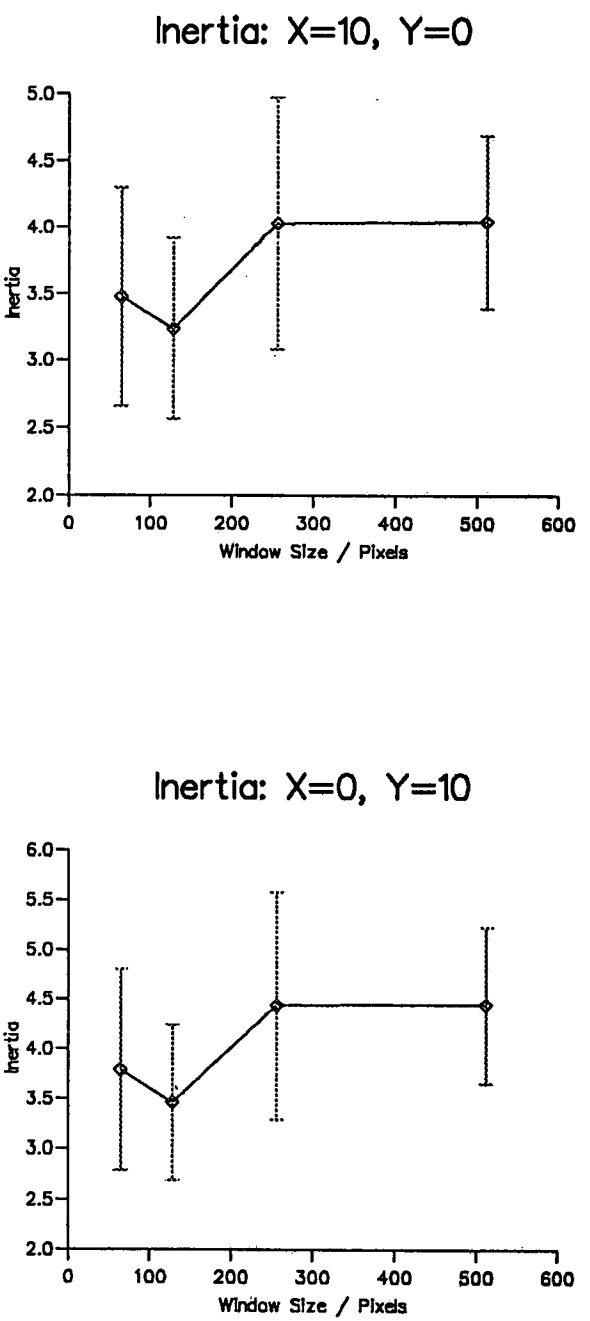

Figure 3.13. Inertia: Locharbriggs Stone. The inertia is shown for the two specified displacements, calculated for various window sizes. 500 pixels = 1 inch (2.54 cm).

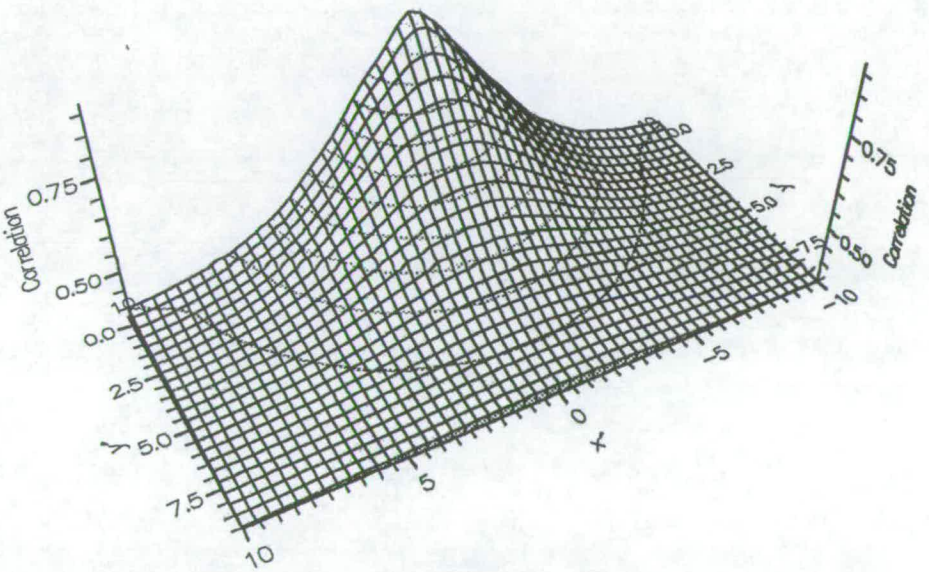

Figure 3.14. The dependence on displacement of correlation for Dunhouse Buff stone is shown. The cooccurrence matrices were calculated for a variety of displacements. The corresponding values of the correlation measure are plotted against these displacements.

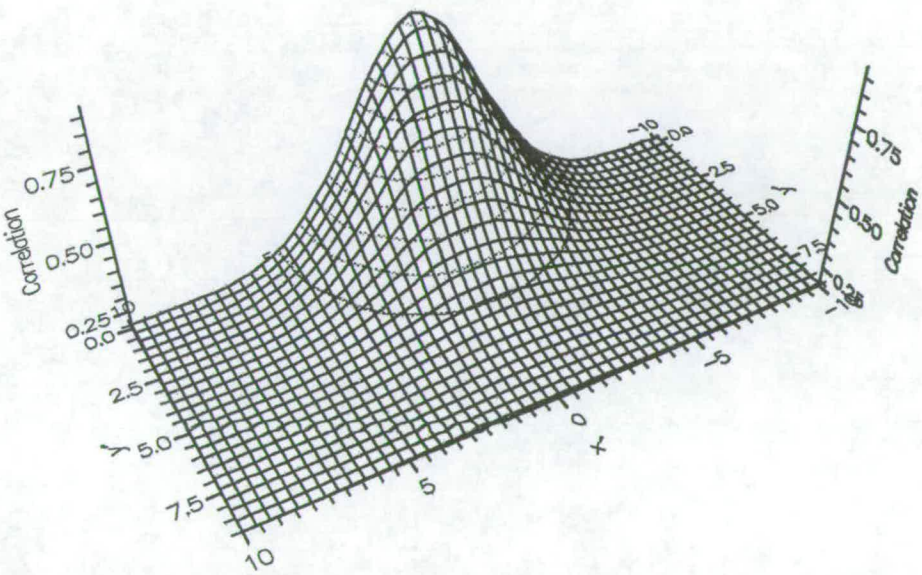

Figure 3.15. The dependence on displacement of correlation for Locharbriggs stone is shown. The cooccurrence matrices were calculated for a variety of displacements. The corresponding values of the correlation measure are plotted against these displacements. For both Dunhouse Buff and Locharbriggs, the dependence is broadly the same, but for the latter the central peak is narrower, suggesting that this rock is less correlated than is Dunhouse Buff.

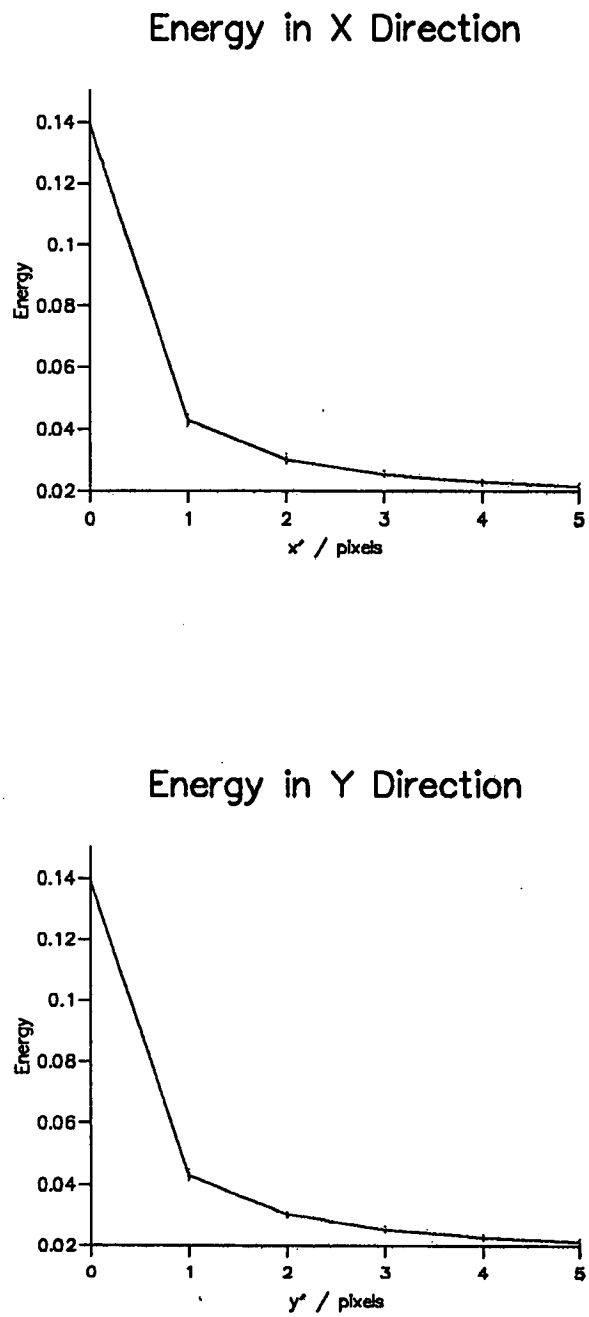

Figure 3.16. Energy measure for an artificial field, which is highly correlated in the y' direction, and anticorrelated in the x' direction.

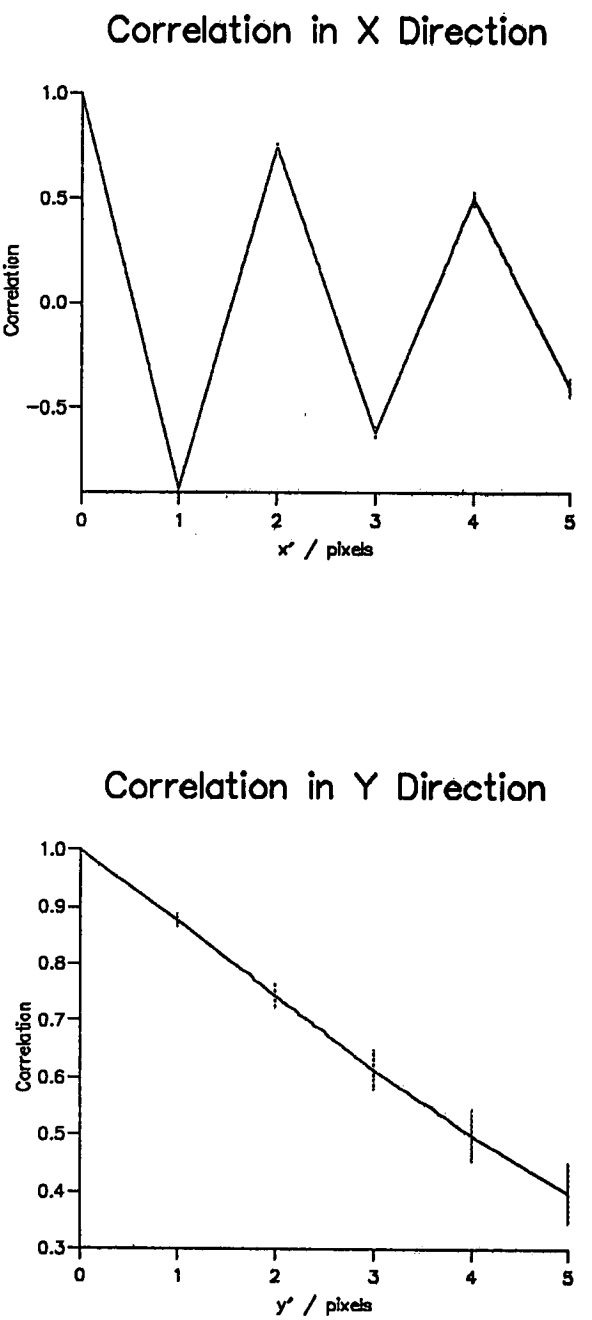

Figure 3.17. Correlation measure for an artificial field, which is highly correlated in the y' direction, and anticorrelated in the x' direction.

Chapter 4

Multilayer Perceptron Training

In Chapter 3 we see how, with appropriate preprocessing, both a simple perceptron and a multilayer perceptron can be trained to distinguish between two types of rock using SGLDM measures as inputs. In general, we need to investigate larger sets of images. This chapter describes the use of the SGLDM measures as inputs to a multilayer perceptron, which is trained to distinguish between several classes of images. There are three aspects to this work: the choice of appropriate parameters for the SGLDM, selection of a suitable network architecture and training algorithm, and the training itself.

This Chapter also describes the use of the K nearest neighbours method for the same purpose as a comparison. The neural network model is found to perform well when compared with this technique.

First, we discuss the images that were gathered, and the way in which they were processed to give data that were used as inputs to the networks.

4.1 Input Data For the MLP

It is necessary to establish a representative selection of images for training the network. In a real application — to which this work can be considered only an approximation — one would prepare a training set containing all the types of images which were expected to occur in the field. In this work, we restrict ourselves to a smaller range of images. The results are only valid within this set of images, though they ought to be replicable in a wider context.

4.1.1 Images Used

Photographic images were collected from the vertically sawn faces of thirteen clastic and carbonate rock types common to hydrocarbon reservoirs. Some of the rock types were chosen from [93] and [94], others from working quarries and one from North Sea core. The limestones are Purbeck Marble (blue), Hopton Wood, Ham Hill (all from [93]) and Tufa. The sandstones are Grenoside, Bramley Fall, Elland Edge Flagrock, Fly Delph Gritstone, Hall Dale and Dunhouse (all from [94]). Three further sandstones are used: Beatrice Core (a bioturbated and oil stained silty-sand), Liesegang Ring (an iron stained sample of medium-grained, subarkosic, fluvial sand) and Mottled Sandstone (a course-grained, arkosic sand with kaolinised feldspar clasts). All of these stones are shown in Appendix B.

The photographs were scanned to produce greyscale images at various pixel densities (600-700 pixels per inch) in such a way that scale changes from one photograph to another were removed.

4.1.2 Data Acquisition and Preprocessing

Preprocessing is necessary to reduce the number of inputs to the network; if there are too many of these for the number of training patterns, the net will merely

"store" the patterns and not generalise from them. The preliminary studies using only two rocks (Chapter 3) confirm that there is not enough image data available to use raw pixel values as inputs; there would be very many inputs, resulting in an over complicated net, slow to train and requiring thousands of training patterns.

The scanned images consist of $L_x \times L_y$ integers $\in [0, 255]$ representing discrete greyscale levels in the photographs. We use an implementation of the greyscale cooccurrence based texture measures (second order statistics) introduced by Haralick[67] and described in §2.3 to extract features from these "raw" images as inputs to the network. Before these features are extracted, however, the images are linearly rescaled so that the pixel values are in $[0, 15]$ with a mean of 7.5. This reduces the computational load and prevents discrimination based solely on lightness/darkness of the images (first order statistics.) The use of a linear rescaling is simpler than the equal probability quantisation described in [67] although the latter may be more appropriate (the effect on the large distance form of the texture measures is discussed below). Time does not permit a comparison between the two techniques. There is a lengthy discussion in [20] of histogram normalisation (the equal probability quantisation used by Haralick *et al* in [67]) and Gaussian normalisation.

We use the same set of texture measures as in the work described in §3.4 — their definitions are given in (2.15) - (2.19). The behaviour of these measures can be understood in terms of the long range order in the image and the general consequences of this are discussed in §2.3.6. These effects are discussed in §5.2.2 and §5.2.3 for textures generated by the Ising model, for which we can calculate the expectation values of the cooccurrence matrix elements P_{ab}. They are shown below (Figures 4.1 - 4.5) for some of the images which we used.

4.1.3 Parameters Used in Calculating Texture Measures

As is discussed in §3.6, two parameters are important in calculating the cooccurrence matrices; window size and displacement (x', y'). A third parameter which might be varied is the "blocking" or "coarse graining" factor. As in Chapter 3, our data is insufficient to do coarse graining and retain enough training patterns. Such problems — restricted quantity of data, limited quality and noise introduced by the data collection process itself — are almost inevitable in real applications.

Window Size (Area)

The area of image with which the cooccurrence matrix is calculated should be large enough to be representative, but no larger. Richards[119, p. 67] uses blocking to find an appropriate length scale, and correlates this with performance of a network trained to discriminate Brodatz textures[15] given input windows of pixels. For periodic patterns it is possible to determine the appropriate scale directly, for instance the unit repeat may be found by plotting $I(x', y')$ [22], but for the more complicated rock images, this is less applicable. On graphs such as Figure 4.5, where the average values of inertia for some of the limestones are plotted against x' (y' is held at 0), I levels off at distances which are much less than the representative elementary length. This is because the cooccurrence matrix based measures are essentially correlation functions. (In Chapter 5 this is shown to be so for the particular case of textures generated by the Ising model. Thus, the measures reflect "local" relations between the pixels.) The representative elementary area may be determined by larger scale effects; this is where a blocking of the image would be useful since it would allow these larger scale effects to be absorbed as inter pixel interactions.

In Chapter 3, a detailed study is described of the effect of sample area (window size) on the cooccurrence matrix and the texture measures. From such a study,

one may obtain, for a given image, some idea of the appropriate area which should be used to identify a given rock type. We are not able to do this for the images in our sample set, because the area of image available is not sufficient. Thus, the area chosen is determined by the amount of image and is a compromise between obtaining as many data sets as possible and retaining as large an area of image as we could. We use the largest window size consistent with producing enough separate data sets for meaningful training. This is 128×128 pixels.

Displacement

Rather than use some particular single value of x' and y' we sample the curves (for $y' = 0$) at 5 locations — $x' = 0, 1, 3, 10, 15$. These values (an approximation to the shape of the measure curves) are the inputs to the network. This gives information about the behaviour of the measures over distance. Figures 4.1, 4.2, 4.3, 4.4 and 4.5 show the values of the measures plotted against both x' and y', in each case with the other coordinate held fixed at 0. The values shown are averages for each image, obtained from separate regions of 128×128 pixels. The curves show that there is no strong dependence on direction in the texture measures. (We show graphs only for the limestones, since there are only four of them. The behaviour of the measures is similar for the sandstones.)

Table 4.1 shows the values of the texture measures at large distance for four different textures. The first two columns show the measures (from (2.30) - (2.34)) for totally random textures, all pixel values being assumed equally probable. In the first column two pixel values ± 1 are allowed. In the second, the pixels may take on 16 values $0, \ldots 15$. The third column shows the expected values for Ising model generated textures, and the fourth the actual ones for Purbeck Marble (Photograph B.11).

The Ising model is a simple statistical physics model of a magnet. It may

Measure	Random Case		Ising Texture	Purbeck Marble
	$N_G = 2$	$N_G = 16$		
E	0.250	0.004	0.250	0.042300
H	0.602	2.408	0.602	3.456405
C	0.000	0.000	0.000	−0.034155
L	0.600	0.293	0.600	0.430758
I	2.000	42.667	2.000	4.352476

Table 4.1. Large distance values for texture measures. The random cases are determined from (2.30) - (2.34), all pixel values being assumed equally likely. The Ising result (for a 1D model, in zero field is an exact result, from the expressions calculated in Chapter 5. It is valid in the limit of $|j - i| \rightarrow \infty (T > T_C)$ or $T \rightarrow \infty (\forall |j - i|)$. The figures for Purbeck Marble are determined from an image, using an input window of 128×128 pixels, with the range of values rescaled from $[0, 255]$ to $[0, 15]$ with a mean of 7.5.

be defined for 1D, 2D, 3D or higher dimensional systems, with or without an external field. Depending on the model used, there may or may not be a critical temperature T_C below which the system is ordered, and above which it is disordered. The model we use here is the 1D Ising model in a zero external field. It has no finite T_C so that the two pixel values ± 1 are equally likely for all $T > 0$; thus the values for the zero field Ising case are the same as those for the random case at large distances. A different result is obtained for some of the measures if pixel values $0, 1$ are used since the values are present in the definitions of some of the measures.

It is clear from these values, the limiting cases discussed in Chapter 2, that the overall behaviour of the texture measures for the rocks agrees qualitatively with theory. At large distances, we assume that all the correlations have decayed away (this will not be true if the image is strongly periodic — see the sinusoidal case discussed in Chapter 2). The texture measures will then be those for random images; (2.30) - (2.34) are derived assuming that all pixel values are equally

likely. However, the large distance values (identical with the random case of (2.30) - (2.34)) should then depend only on N_G, which is obviously not the case (see Figures 4.1 - 4.5) since they are different for different rocks. Clearly, the equal probability assumption is not true. For reasons of time and simplicity, we simply did a linear rescaling on our image data, fixing the mean at 7.5 (as well as reducing the number of pixel values to 16 from 256). If this rescaling removes differences in tone which could otherwise be used as recognition cues, then the texture measures reflect real textural differences which can be used to discriminate images. If, however, a linear rescaling leaves such differences in tone intact, then it is possible that we are using tonal differences to discriminate the images.

The rescaling reduces the overall variation in tone between the data samples. Richards[119] takes raw pixel values as network inputs without doing any image rescaling or normalisation. Conradsen[20], on the other hand, performs extensive image filtering and a histogram normalisation which he compares with a Gaussian normalisation of the image. Both of these studies were concerned with the discrimination of Brodatz textures, but the differences between them are considerable. A large part of [119] is devoted to the development of the Rhwydwaith simulator, a neural network package. [20] is motivated by industrial applications of texture recognition. The approaches of the two authors are perhaps representative of the physics and the image processing communities.

4.2 Training

Several parameters are involved in the training, some of which were fixed and some of which were varied so that the effect of them upon training could be studied. In this section, we discuss the parameter selection, the training, and the

results.

4.2.1 Software and Hardware Used

Some investigations were made on the Neurosys neural network simulator running on the Edinburgh Concurrent Supercomputer (ECS). It was found that this simulator is not efficient for our problem, because the networks used are too small. A multiprocessor machine, such as the ECS, has many advantages over a conventional one but it incurs communications overheads. Unless a problem is large enough to give economies of scale these overheads cause slowing down. In addition, the Neurosys package is still under development and (at the time of writing) it will only perform backpropagation. Chapter 2 showed that there are alternative algorithms which often run faster than backpropagation. Ideally, one would wish to have several different schemes available to suit different problems.

We chose to use the Quickprop algorithm[39] which is described in Chapter 2. It is broadly similar to backpropagation but has been shown to run faster for a variety of problems[37]. We obtained the NevProp software package[1] which we ran on various Sun SPARCclassic workstations and, later, on a Sun Sparc Centre 2000 (6 10/41 processors).

Once the hardware and software are chosen, it is necessary to determine parameters for use in the training.

4.2.2 Constant Parameters

We use constant values of some parameters: weight decay is 0.001, momentum is 0 and weights are initialised randomly in $[-1, +1]$. All updates are done in batch mode, modifying the weights after presentation of the entire pattern set;

[1] Developed by Phil Goodman, David Rosen and Allen Plummer of the University of Nevada Center for Biomedical Modeling Research.

to prevent the weight modification from being too great in any one epoch, the update step was normalised by the number of input nodes. We use an atanh transfer function rather than a sigmoid.

The "threshold" for correct scoring of outputs is 0.5; with outputs 0 or 1 this means that any single output has a 50% chance of being correct by chance. It also means that a small perturbation of the weights would reverse the classification of borderline outputs. Krogh and Hertz[89] suggest that the threshold should be less than 0.5 to prevent this, but since we are ultimately concerned with scoring patterns rather than individual outputs, we do not consider this a problem.

4.2.3 Training Set

The inputs used (§4.1) are values of the 5 texture measures for 5 values of x' (and y' fixed at 0) giving 25 inputs in all. The outputs are represented by 4 bit binary vectors giving 16 possible output patterns, 13 of which were allocated to the known classes. The remaining 3 are unused; when testing networks, any output corresponding to one of these "unused" patterns is regarded as being wrong. (One could classify them as "unknown" which would improve the success rate slightly.)

A training set of 500 input/output pairs was constructed by selecting random examples from the set of 2500 patterns. The remaining 2000 patterns are used to test the network. The same training and test sets are used each time. Table 4.2 gives the proportion of each class in the training and in the test sets. It might be argued that one should choose equal numbers of all the different classes in the training set; but since there are more patterns in some classes, we feel that better use can be made of them by drawing patterns at random. Thus, the proportions of each class in the training and test sets are approximately equal. Examination of the results (Table 4.3) does not suggest that the most frequent patterns are

scored correctly more often than the less frequent ones.

Rock Type	Training Set	Test Set
Grenoside	0.046	0.051
Bramley Fall	0.083	0.072
Elland Edge Flagrock	0.053	0.047
Fly Delph Gritstone	0.095	0.100
Hall Dale	0.051	0.049
Dunhouse	0.107	0.095
Purbeck Marble	0.095	0.098
Hoptonwood	0.049	0.050
Ham Hill	0.096	0.100
Tufa	0.090	0.098
Beatrice Core	0.043	0.047
Mottled Sandstone	0.102	0.100
Liesegang Ring	0.090	0.092

Table 4.2. Composition of training and test sets — fraction of the training and test sets made up of each class of pattern.

4.2.4 Error Measures

Two types of error are considered: **per pattern** and **per output** errors. The former is based on a more stringent criterion than the latter; for a pattern to be scored correctly, all of its bits must be correct (to within the specified tolerance). Each pattern consists of 4 outputs. When the results are thresholded then an output has a success probability of 0.5, given a random choice of weights; a pattern has a probability of $(0.5)^4 = 0.0625$ of being correct by chance.

It is the per output error which is monitored by the NevProp software during training. Thus, we present graphs of per output error against training epoch. We wrote further software in order to determine the per pattern error from a file of predictions made by NevProp. Per pattern error is only calculated for the end of particular runs of training.

4.2.5 Training Procedure

Network training was repeated 10 times for each combination of parameters — the fewest number of repetitions allowing an estimate of error. Up to 15000 epochs of training were allowed in the network runs which investigated these varying parameter regimes, though NevProp could break off the training if learning was poor. At least 200 epochs of training were required before the program was allowed to terminate; before terminating, the program was also constrained to run for 4 times longer than the "best" epoch. Both of these strategies were to avoid local minima. One sign of poor parameter regimes was the large number of runs which terminated well before the allowed 15000 runs were done.

Graphs are presented to show the dependence of the fraction of successfully classified patterns on the various parameters. Other graphs show the evolution of successful output classifications over training for various selected regimes.

4.2.6 Variable Parameters

The effect of varying the following parameters was investigated, and is discussed below: stepsize ϵ, the number of hidden layer units h, and noise on the weights at update (this was an addition which we made to the code.) We consider h first, then ϵ and finally noise.

Network Architecture

Because the optimum number of hidden layer nodes is not known, it is necessary to try different network architectures. With 25 inputs, (the values of the 5 selected texture measures for $y' = 0$ and $x' = 0, 1, 3, 10, 15$), one hidden layer with h nodes, and 4 outputs, and full connectivity between the nodes in adjacent layers, the various networks investigated can be represented by a notation `25:h:4` and they

are parameterised solely by h.

We use only one hidden layer because it is possible in principle for this one layer to represent any continuous function [32, 74], §2.1.2. In addition, the more hidden layers are used, the more time consuming is the training, since it has to be repeated for many more networks to find the best one. Additional variables are also introduced — the number of nodes in each layer and the number of layers — requiring more training patterns.

The upper graph in Figure 4.6 shows the effect of various numbers of hidden nodes on the final per pattern success (the fraction of patterns in the test and in the training set which the net correctly classifies). The lower graph shows how the per output success evolves through the training, for a variety of network architectures with different numbers of hidden nodes. It is clear that as the number of hidden nodes increases, performance gets better until a limit is reached.

Update Size ϵ

Figure 4.7 shows the effect of varying ϵ on the per pattern success for a fixed architecture 25:20:4 with no noise. For ϵ too large, performance is poor; as ϵ falls, performance improves and then begins to fall again though further training would be necessary to confirm the trend.

Table 4.3 summarises these results, and shows how the different types of rock are classified for each value of ϵ.

Noise

It is often useful to train a neural network with noise; this either means adding noise to the data, (making more data sets available; or allowing for the effect of corruption on future input data) or interfering with the weights by adding a small random number, drawn perhaps from a Gaussian distribution, to the

weights after the update. For simple nets, it is possible to calculate the effects of such a procedure on the generalisation and training errors.

The purpose of adding noise to the weights during training is to improve generalisation. This is essentially similar to other techniques for avoiding local minima in the error surface (such as momentum or weight decay). The noise lifts the network out of such minima.

We add a small noise parameter to the weights after update, drawn from a Gaussian with a known variance centred on zero. In the upper graph of Figure 4.8, the number of successfully classified patterns, for both the test and the training sets, is plotted against the variance of this distribution. The simulations were repeated for various values of ϵ.

We now turn to a discussion of the results of this work.

4.3 Results

4.3.1 Architecture

It is apparent that when the net is too small, performance suffers. Such a net is incapable of representing the classifications which one requires it to learn. Conversely, a net that is too large will **store** training patterns without generalising well. This is starting to happen with `25:35:4`. Somewhere between these extremes lies an ideal architecture (see §2.1.2 and [89, p. 142]). There is an often quoted rule of thumb in neural network research that a network can store up to approximately mh patterns, where h is the number of hidden nodes and m the number of inputs. For 500 training patterns, $h \approx 20$. The results are consistent with this.

Rock Type	ϵ			
	10^{-3}	10^{-4}	10^{-5}	10^{-6}
Grenoside	98	96	95	97
Bramley Fall	39	49	51	49
Elland Edge Flagrock	37	37	34	40
Fly Delph Gritstone	49	53	55	54
Hall Dale	42	43	48	46
Dunhouse	68	68	69	48
Purbeck Marble	75	78	79	76
Hoptonwood	17	25	25	22
Ham Hill	51	59	56	55
Tufa	97	95	94	94
Beatrice Core	62	77	83	73
Mottled Sandstone	81	81	81	80
Liesegang Ring	76	79	79	79
Overall Success	64±2	67±2	68±2	66±3

Table 4.3. Training Results. The percentage of test patterns correctly classified for each rock type and for various values of ϵ. Overall success rates are also given. Note the especially bad results for Hoptonwood — even though this stone is visually distinct. This illustrates the general poor correlation between the appearance of the rocks and the ease with which the network classified them correctly.

4.3.2 Update Stepsize ϵ

In Figure 4.7, the training and the test success are plotted against ϵ. We wished to use as large a value of ϵ as possible in order to make training fast. (A larger stepsize should correspond to faster training). It is clear that when the value of ϵ is too large, poor performance results. As ϵ is reduced, the performance improves but not significantly.

Although we did repeat these simulations for a variety of larger ϵ the overall effect was always similar — increasing noise degrades performance. In these studies, we used $h = 20$ because it was apparent both from theoretical and observational considerations that this was close to the optimum number of hidden

nodes.

If ϵ were reduced too much, training time would be prolonged (this effect is not obvious in our results because we were training to a maximum number of epochs).

4.3.3 Noise

The effect of noise may also be seen in the lower graph of Figure 4.8 where for constant ϵ and h different noise levels are used. The test set output success is plotted against the number of epochs of training for a variety of values of the noise variance. It is clear that as the noise increases, performance over time becomes worse, and eventually performance begins to worsen over time. (Note that success is initially 50% as one expects for the per output success).

There is no apparent benefit in using noise here; as it increases, both generalisation and training error become worse (though the difference between them decreases, so relatively speaking, the generalisation has improved). The lack of success using noise was surprising since we were performing batch training. If updates are done "on line" and the patterns presented in random order, noise is generated by the randomisation and ought not to be required. Batch updating risks the weights getting caught in local minima when weight updates cancel and it ought to be aided by the use of noise[89, p. 129].

There may be several reasons for the lack of success using noise. It is possible that we have not discovered an appropriate parameter regime; most likely, the noise is too large — for some much smaller value, performance might improve again. To find this would require more extensive simulation.

4.4 Nature of the Texture Measures.

Table 4.3 shows that a neural network, presented with the quantities energy, entropy, correlation, local homogeneity and inertia, is able to classify the images from which these quantities were derived with a reasonable accuracy. No training was done using subsets of these quantities, to see whether or not the net would be able to classify images based (for instance) on energy alone. There must be strong grounds for thinking that this would be so. See, for instance, equations (5.38 - 5.42) where an exact form is calculated for the five measures for textures generated by a particular model (the 1D Ising model in zero field). The form agrees qualitatively with Figures 4.1 — 4.5 which show the actual measures for the limestones.

Equations (5.38 - 5.42) show that, for the 1D Ising model, the texture measures are all simple functions of the correlation

$$\left(\frac{\lambda_1}{\lambda_0}\right)^{(j-i)}$$

where λ_0 and λ_1 are the eigenvalues of a matrix and $(j-i)$ is the distance between the sites on the Ising chain. We may well claim, then, that it is only this basic quantity which is being presented to the network. All of the other measures are redundant, and the net should be able to distinguish between the rocks based on correlation alone.

Of course, it is possible that one or more of the various functions of the correlation is more suited to represent the essential nature of the different rock textures. We mentioned in Chapter 2 the problem of choosing an appropriate internal representation, using an example from [89]. It is known that the choice of such a representation can be important; but we do not believe that it makes a great deal of difference where the network inputs are such simple functions as equations (5.38 - 5.42). At worst, the need for the net to produce its own internal

representation might slow it down slightly in learning the inputs; there would be no appreciable difference in accuracy when identifying unseen patterns.

To summarise, the fact that one potential network input is a modification of another does not mean that it is equivalent as an input. It may be better, or worse (where these descriptions are related to how useful the input is in helping the net to distinguish between different classes of input patterns). The art of training a network requires some skill in choosing a good input representation for the problem in question; this is emphasised in most of the standard textbooks. However, in this particular case we do not believe that there was any significant advantage or disadvantage in the texture measures compared to the base quantity, correlation. This means that using additional inputs may have slowed down the training, and meant a more complicated network structure than would otherwise have done the job. But a detailed study of the actual weights evolved ought to show that the redundant inputs were ignored; that is, their weights were driven to zero.

4.5 K Nearest Neighbours Method

In order to set the results of neural network training given above into context, it is important to compare them with an established statistical procedure. We see in Chapter 2 that statistical classifiers rival neural networks for many purposes. We use the method of K nearest neighbours[87, 10] because it is simple to code, non-parametric and easy to understand, yet has a basis in theory. It is also suited to the use of the data sets to hand, which have already been used for network training, so that comparisons can be made.

4.5.1 Algorithm

We are given a "training" set of M vectors ξ, whose output classes are known, and a test set of N vectors ρ, whose output classes we wish to infer. For a given test vector ρ', calculate the difference

$$s^2 = \sum_i (\xi_i - \rho'_i)^2$$

for each vector in the training set. From the M values obtained, select the K smallest ones (the "nearest neighbours") and determine which output class occurs most frequently among them. Assign the test vector to this class. This procedure leaves open the question of what is the most suitable K value.

4.5.2 Results

Figure 4.9 shows the results for a K nearest neighbours classification using the same training and test sets as for the neural networks described above. The most significant curve is the solid line, representing the "generalisation capability". The dotted line represents the wrongly classified patterns, and the dashed line the unclassified ones. The "unclassified" category was selected when two or more different classes had the same number of members within the K nearest. This was only possible for $K > 1$. As K increases, the chance of such a "tie" happening falls. How should we treat such cases? Theory suggests that the corresponding classes are equally likely, so perhaps one ought to assign the vector randomly between them. This would increase both the success rate and the error rate. Alternatively, one could perform all the assignments probabilistically.

Performance is significantly worse than for the neural networks (the best success rate for K nearest neighbours is 59% compared to 68% for the network) whichever value of K is chosen; on the other hand, the K nearest neighbours method does not require lengthy training.

4.6 Missclassifications

It is interesting to relate the performance of the neural network to the actual images which were being discriminated, and the measures that were calculated from them. This can be done using "confusion tables" showing how the images were (miss)classified by the network (Table 4.4), which images are considered similar by a (non geologically knowledgable) human (Table 4.5) and also (Table 4.6) which images have similar texture measures.

In the following discussion, we do not consider the confusion of the measure curves (Table 4.6). With five different measures, there are too many different curves for a meaningful comparison. Table 4.6 shows our judgement of which rocks had measures which were most alike, but it is given only for completeness.

We find that the rocks confused least by the network were Tufa and Beatrice Core. Tufa is visually distinct, and is not confused by the network with the other rocks. Beatrice Core is visually confused with Elland Edge Flagrock, Dunhouse Buff and Liesegang Ring. The network confused Mottled Sandstone and Liesegang Ring with each other (our volunteer did not) but but not with any other rocks, while the volunteer found it difficult to distinguish Mottled Sandstone from Bramley Fall, Fly Delph or Hall Dale, all of which are relatively coarse grained stones.

The network had most difficulty with Bramley Fall, Elland Edge, Grenoside and Hoptonwood, each of which was confused with five of more others.

Visually, the images can be grouped into four categories. (We emphasis that these are not geologically motivated descriptions, but just visual impressions.)

Coarse rocks Bramely Fall, Fly Delph, Hall Dale and Mottled Sandstone.

Unstructured rocks Elland Edge, Dunhouse Buff, Beatrice Core and Liesegang.

Marbled rocks Purbeck Marble and Hoptonwood

Miscellaneous rocks Ham Hill, Tufa and Grenoside. (All visually distinct.)

This classification is reflected in the lack of confusion betweeen the rocks in the miscellaneous class. The coarse rocks tend to be confused, as do those in the unstructured and marbled classes.

Turning to the behaviour of the neural network, we find no such clear picture. Bramley Fall is confused with several others, as is Elland Edge, including (respectively) some coarse and some unstructured rocks. Both are also confused, however, with other rocks, including Hoptonwood and Purbeck Marble in both cases, which are members of a very different group.

4.7 Conclusions

This work shows that some of the cooccurrence based measures introduced by Haralick[67], when plotted against their (x', y') parameters, give curves which can be used by a neural network architecture to recognise lithofacies types based on image input. The trained network has a recognition rate (up to 68%) which improves upon that of a very simple statistical classifier.

There is an optimum architecture for the network, which gives the best recognition. This is a two layer network with 20 hidden units. The dependence upon training parameters is generally in qualitative agreement with theoretical expectations. Uncertainty remains about the most appropriate parameters, but extensive studies comparing different parameter regimes, with an averaging over training sets, would enable these to be fixed.

It would be useful to repeat this work using different randomly chosen training and independent test sets (cross validation) so that the results are averages over the space of possible training sets. This would avoid effects peculiar to a particular training set. It would also be informative to use different sizes of test and training

sets.

Study of the missclassifications suggests that, at the least, there is no clear relation between the type of missclassification made by the neural network and the type made by a human volunteer. It would appear from this that the features being used are not identical. More study could be done to reduce the size of the pattern vector used by the network, and to select a suitable combination of measures that would reproduce the pattern of human missclassifications. Such work falls within the field of investigating human texture recognition rather than the practical task of designing a system to classify real world textures.

In the next chapter, the cooccurrence matrices will be examined in the framework of statistical mechanics.

	Grenoside	Bramley	Elland	Fly	Hall	Dunhouse	Purbeck	Hoptonwood	Ham	Tufa	Beatrice	Mottled	Liesegang
Grenoside	-	*	*		*				*				
Bramley	*	-	*	*	*	*	*	*					
Elland	*	*	-		*	*	*	*					
Fly		*		-				*	*				
Hall	*	*	*		-	*							
Dunhouse		*	*		*	-		*					
Purbeck		*	*				-	*					
Hoptonwood		*	*	*		*	*	-					
Ham	*			*					-				
Tufa										-			
Beatrice											-		
Mottled												-	*
Liesegang												*	-

Table 4.4. Confusion Table for Neural Network. This table shows which pairs of textures are missclassified by the network. Where the proportion of missclassifications is greater than 0.05 a * is shown. This table has been made symmetric. $\epsilon = 10^{-5}$.

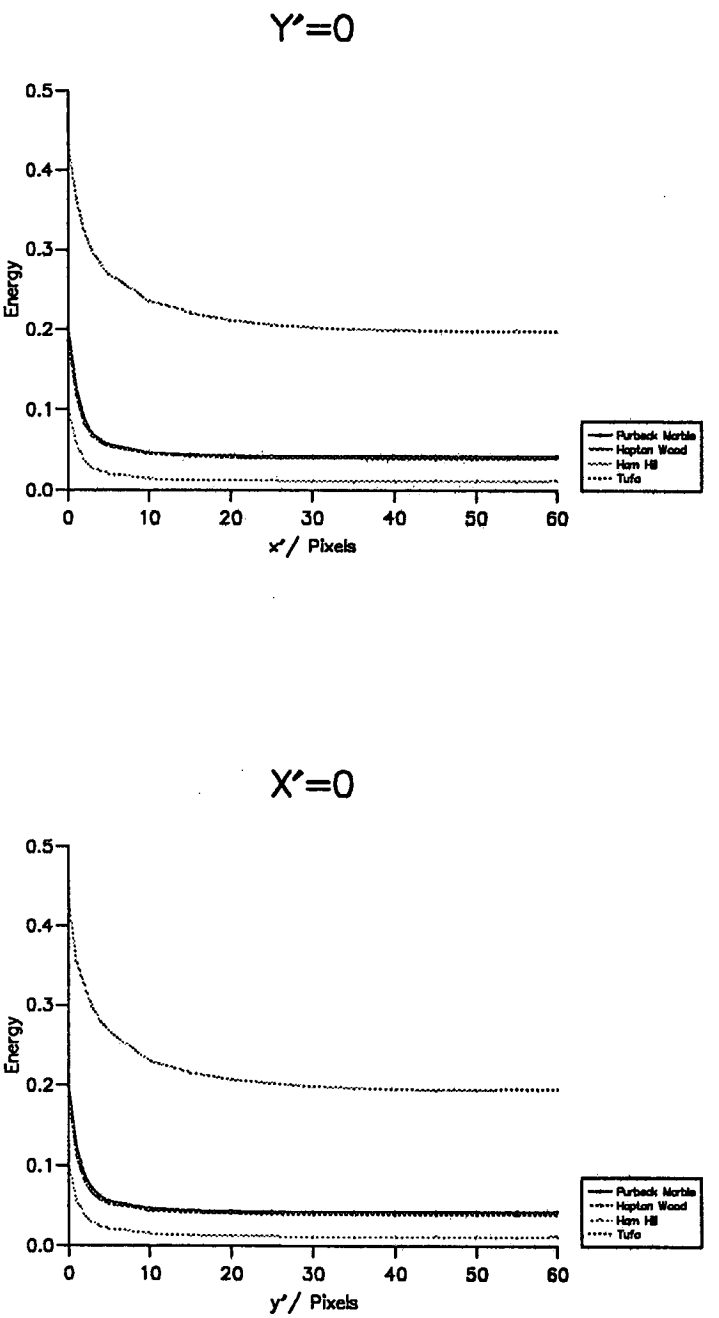

Figure 4.1. Energy Measure for Limestones. The energy is plotted against displacement in both the x' and y' directions, the value of the other coordinate being held at zero.

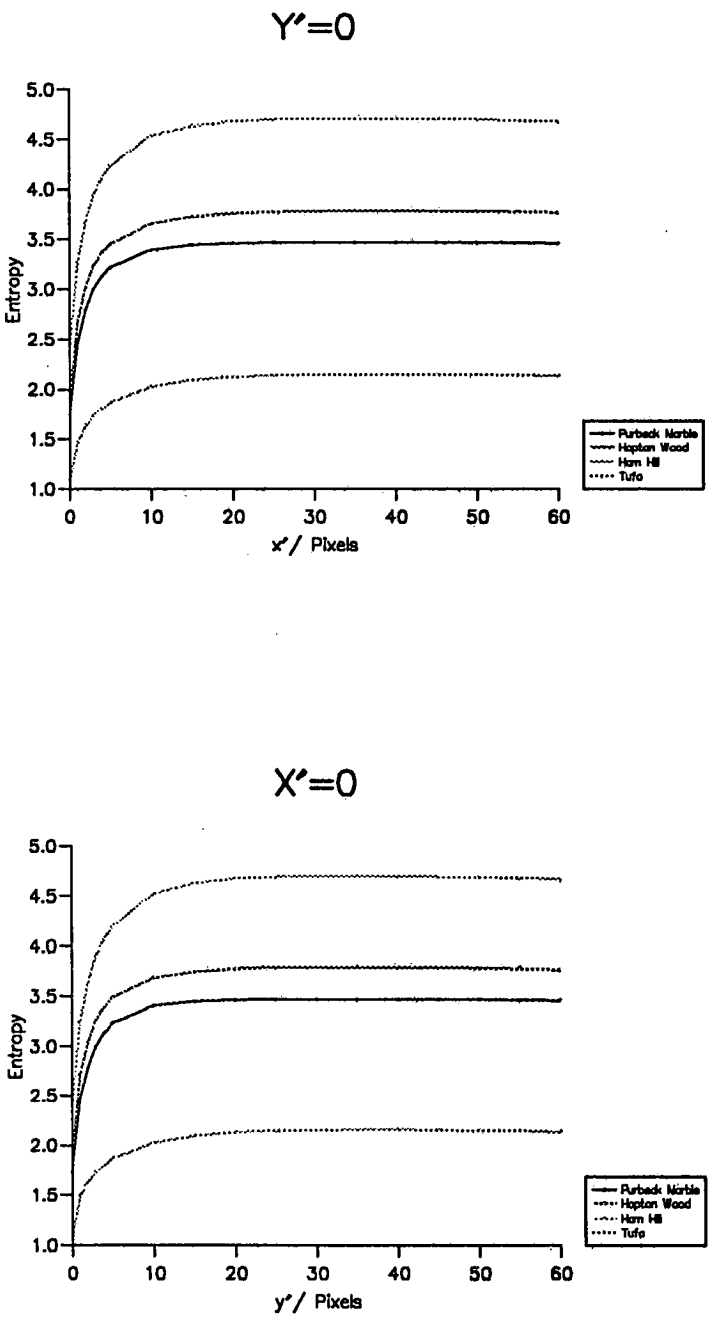

Figure 4.2. Entropy Measure for Limestones. The entropy is plotted against displacement in both the x' and y' directions, the value of the other coordinate being held at zero.

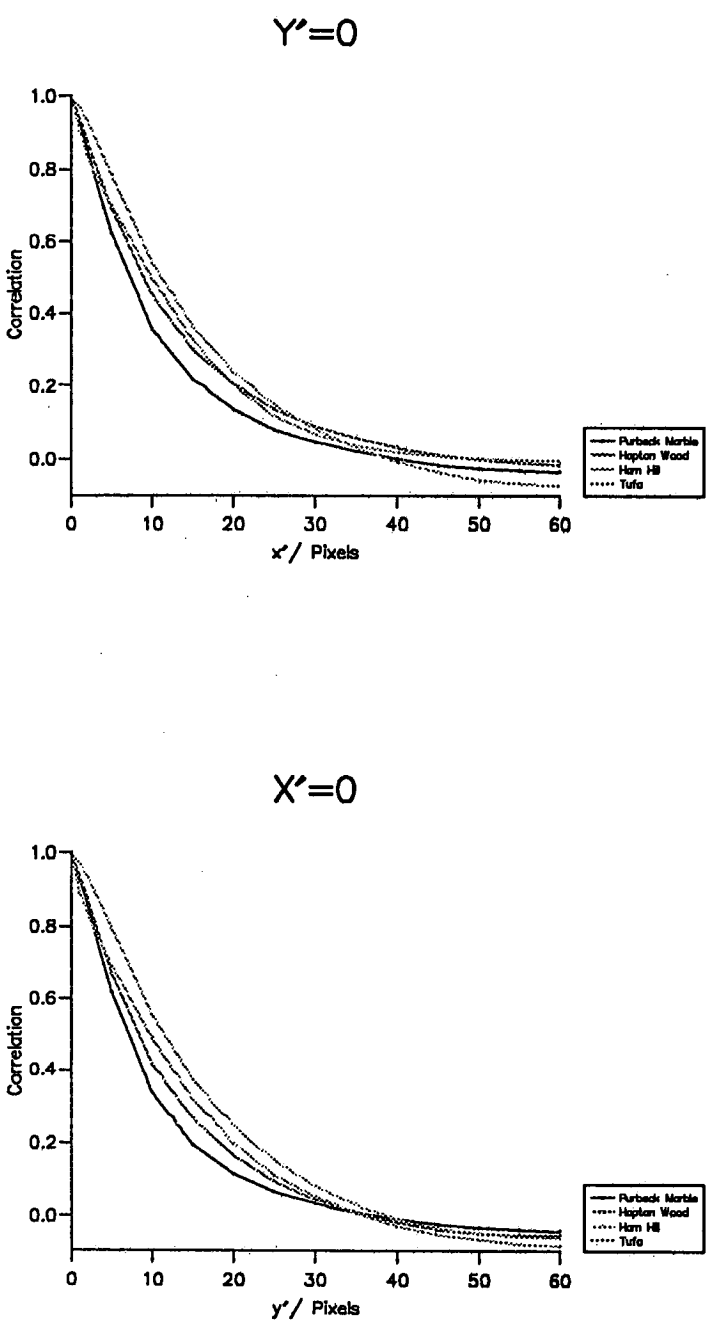

Figure 4.3. Correlation Measure for Limestones. The correlation is plotted against displacement in both the x' and y' directions, the value of the other coordinate being held at zero.

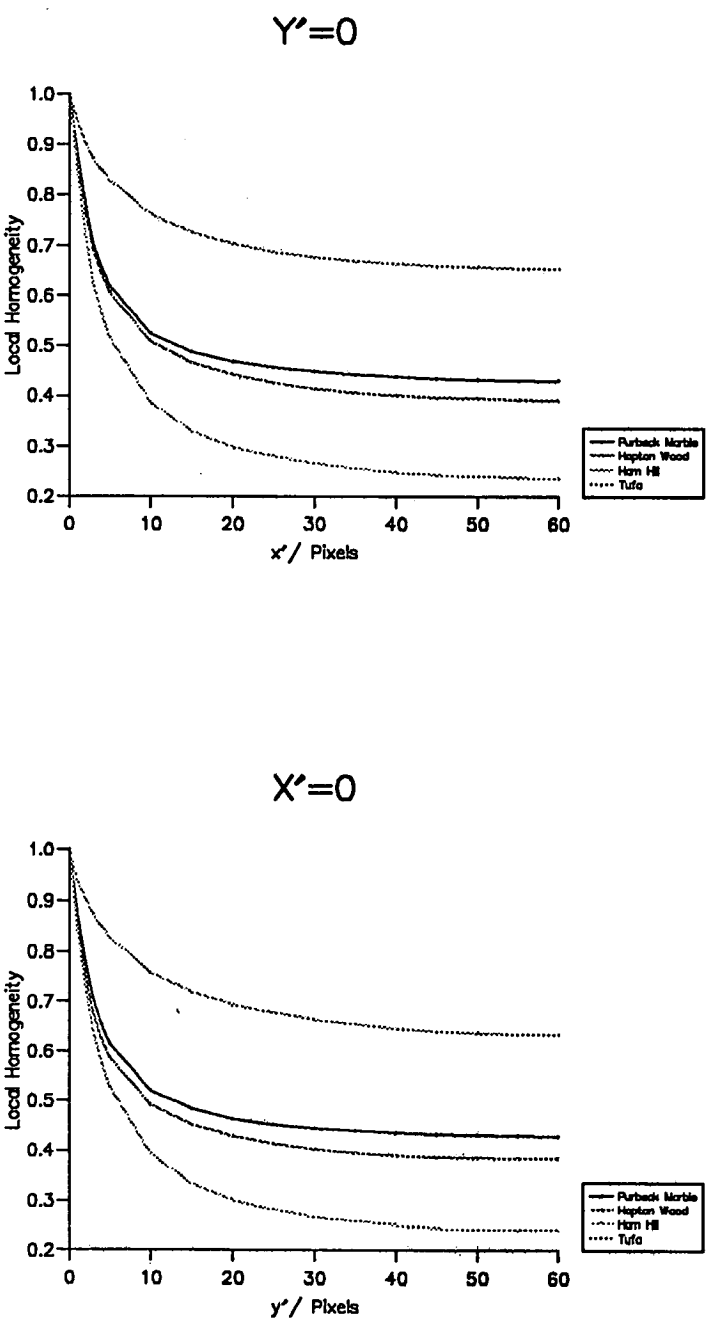

Figure 4.4. Local Homogeneity Measure for Limestones. The local homogeneity is plotted against displacement in both the x' and y' directions, the value of the other coordinate being held at zero.

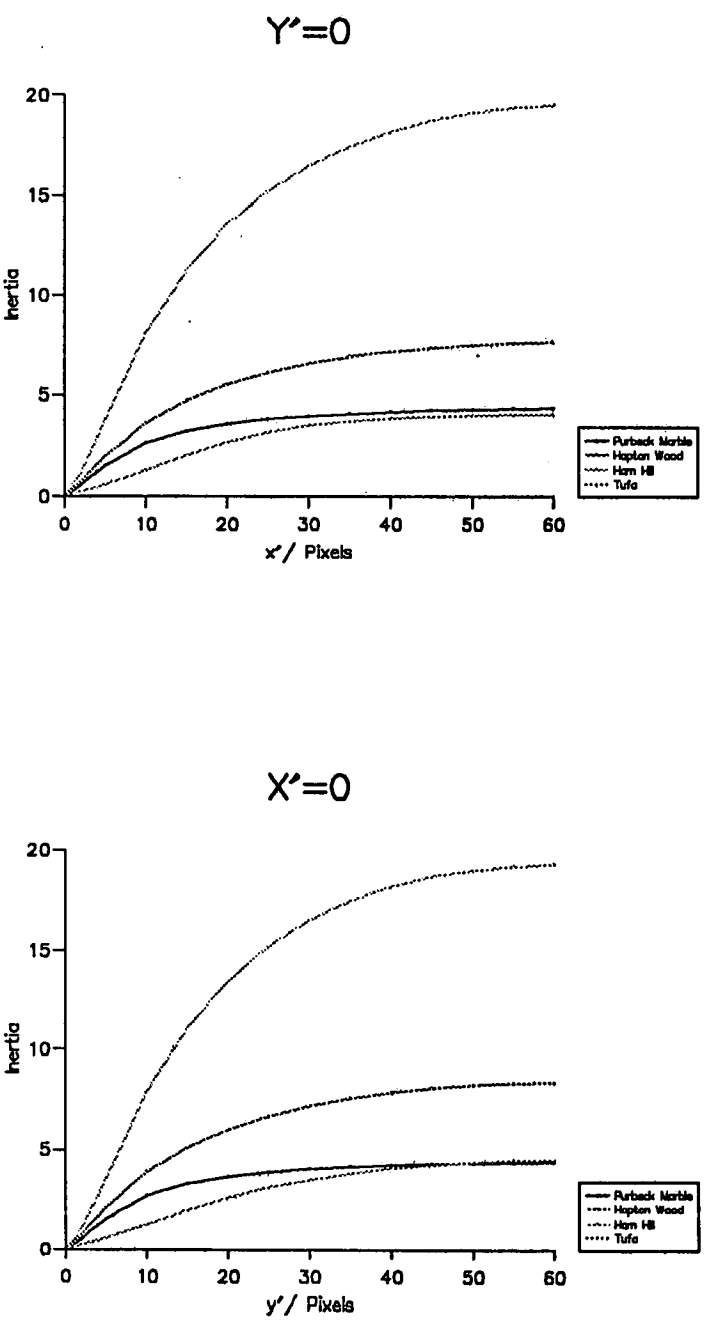

Figure 4.5. Inertia Measure for Limestones. The inertia is plotted against displacement in both the x' and y' directions, the value of the other coordinate being held at zero.

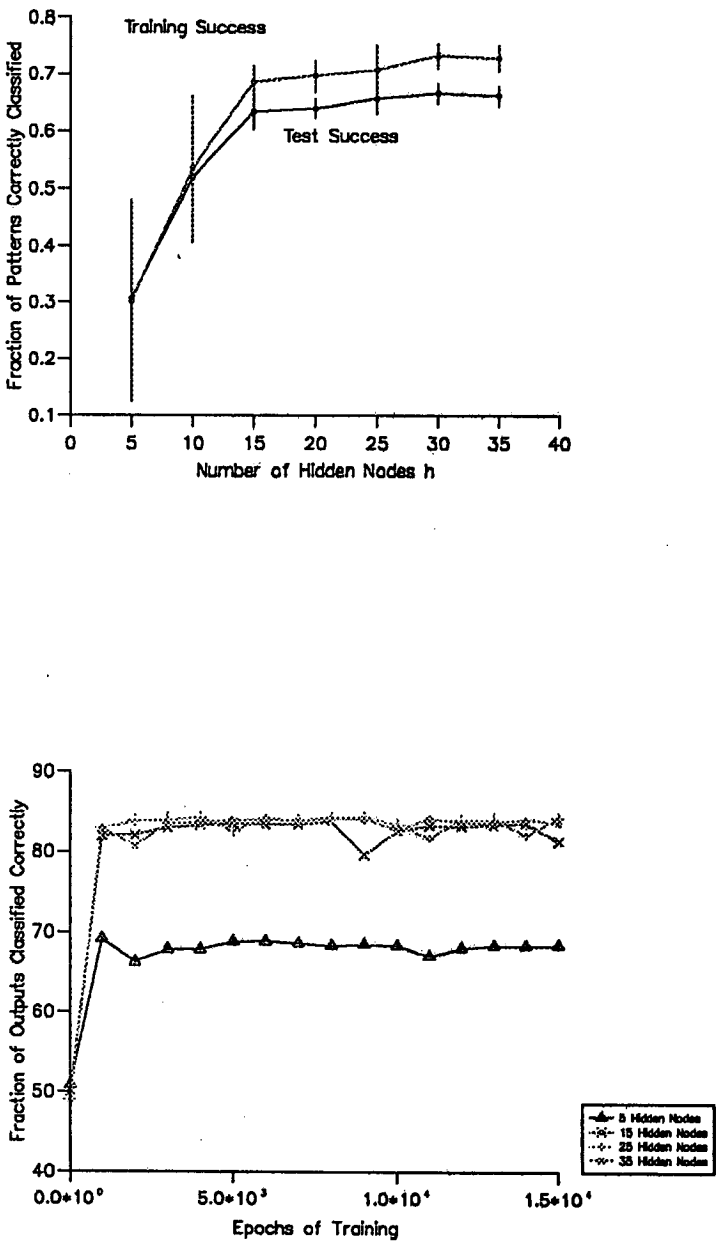

Figure 4.6. Effect of hidden layer size on training. The upper graph shows the per pattern success plotted against the number of hidden layer nodes h. The lower curve shows the per output test success plotted against the number of epochs of training for various numbers of hidden layer nodes. $\epsilon = 0.001$, Noise $= 0.00$.

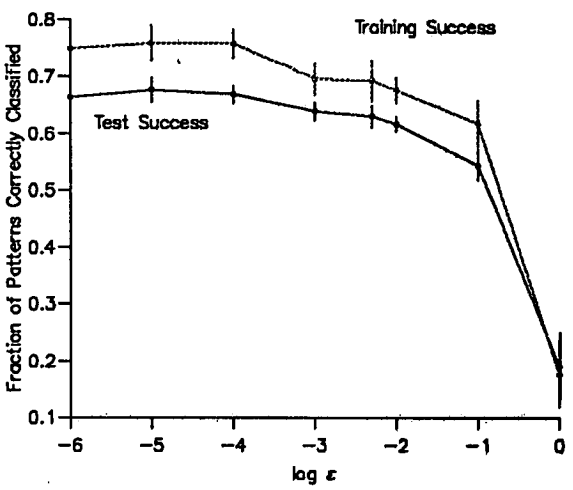

Figure 4.7. Effect of update size ϵ on training. The graph shows the per pattern success plotted against ϵ for a fixed architecture with 20 hidden nodes and noise = 0.00.

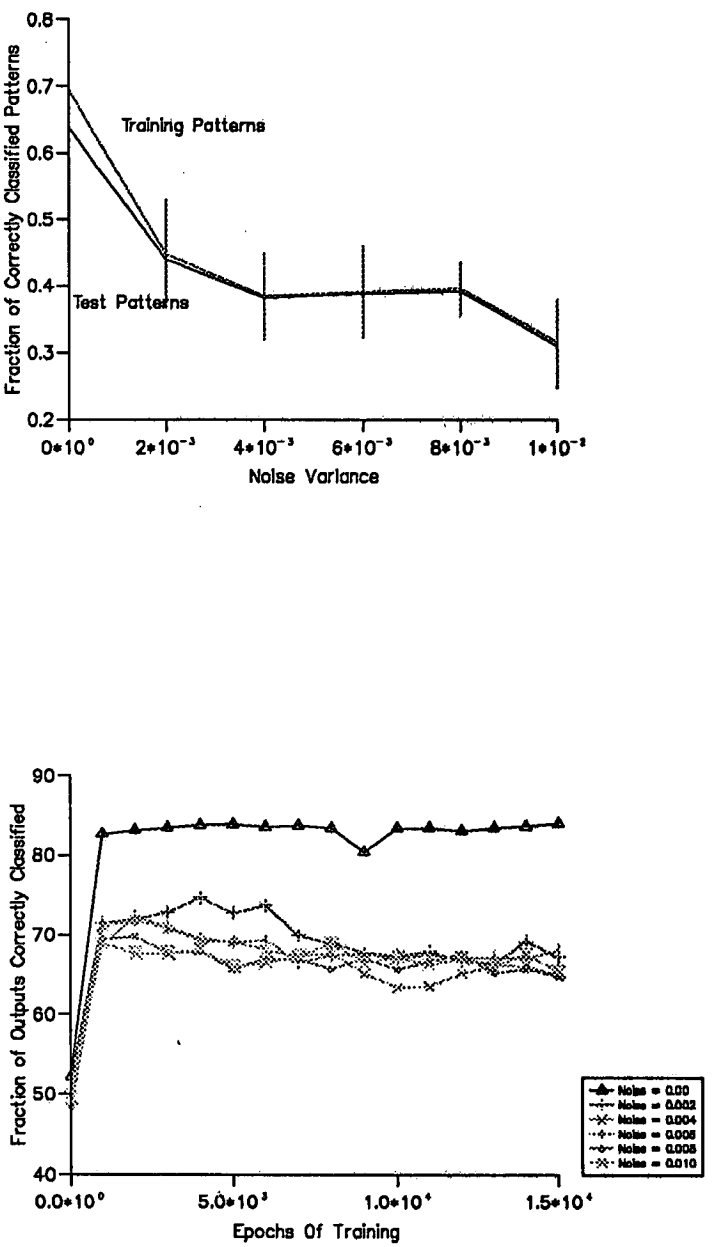

Figure 4.8. Effect of noise on training. The upper graph shows per pattern success of the trained network, plotted against the noise variance. The lower graph shows the per output test success plotted against the number of epochs of training. Each curve represents a different value of noise. $\epsilon = 0.001, h = 20$

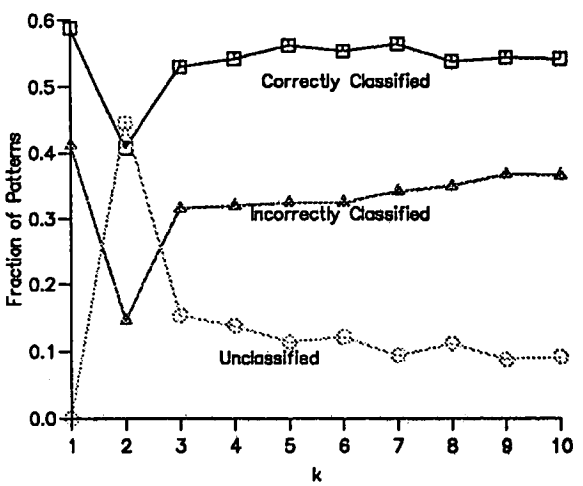

Figure 4.9. Result of K nearest neighbours classification using the same training and test sets as were employed for neural network training.

	Grenoside	Bramley	Elland	Fly	Hall	Dunhouse	Purbeck	Hoptonwood	Ham	Tufa	Beatrice	Mottled	Liesegang
Grenoside	-												
Bramley		-		*	*							*	
Elland			-			*					*		
Fly		*		-	*							*	
Hall		*		*	-							*	
Dunhouse			*			-					*		*
Purbeck							-	*					
Hoptonwood							*	-					
Ham									-				
Tufa										-			
Beatrice			*			*					-		*
Mottled		*		*	*							-	
Liesegang						*					*		-

Table 4.5. Confusion Table for Images. The 13 images were sorted by a human volunteer who identified pairs of images which were confusable. The images used were greyscale copies similar to those in Appendix B.

	Grenoside	Bramley	Elland	Fly	Hall	Dunhouse	Purbeck	Hoptonwood	Ham	Tufa	Beatrice	Mottled	Liesegang
Grenoside													
Bramley				1	1	1					1		1
Elland						2					2		
Fly		1			1	1					1		1
Hall		1		1		1			1		2		1
Dunhouse		1	2	1	1			1	1		3		2
Purbeck								3	1	1			
Hoptonwood						1	3		1	1			1
Ham					1	1	1	1					
Tufa							1	1					
Beatrice		1	2	1	2	3							1
Mottled													
Liesegang		1		1	1	2		1			1		

Table 4.6. Confusion Table for Measures Curves. For each pair of texture measures, the graphs showing the measures plotted against x' were compared. Where the curves were not clearly distinct, they were held to be confusable. The table above shows how many of the texture measures were confusable for each pair of rocks. For example, the table suggests that Dunhouse and Beatrice are easily confusable. A blank entry indicates that no measures were confusable.

Chapter 5

SGLDM and Statistical Mechanics

In this chapter, we consider the cooccurrence matrices (and texture measures) in the framework of statistical mechanics. In particular, we calculate the expectation values of the cooccurrence matrix elements for some simple models. We also consider briefly how textures might be regenerated given the cooccurrence matrices.

5.1 Statistical Mechanics

This section gives a brief outline of statistical mechanics, which will be drawn upon later in the chapter, where we use statistical mechanics techniques and concepts to investigate the cooccurrence matrices. First, we define some terms and establish a formalism.

5.1.1 Formalism

In statistical mechanics we calculate the expectation values of various quantities defined on a system. We work with systems which have many degrees of freedom. States of such systems can be specified as configurations; in this chapter we restrict our attention to systems composed of many interacting spins so that a configuration is merely a set of values for these spins, which are regarded as being placed on a lattice. We write such a configuration $\{S\}$.

Associated with a configuration $\{S\}$ is an energy $E(\{S\})$. A central principle of equilibrium statistical mechanics is that the probability $P\{S\}$ of the system being found in a particular configuration $\{S\}$ is proportional to

$$e^{\frac{-E(\{S\})}{kT}} \tag{5.1}$$

where k is Boltzman's constant and T is temperature. This is discussed further in [43]. The normalisation for (5.1) is called the partition function, and the symbol Z is normally used to denote it.

$$Z = \sum_{\{S\}} e^{\frac{-E(\{S\})}{kT}} \tag{5.2}$$

Given the probability (5.1), one can calculate the expected values of observables G, defined as functions of the spins, if the energy associated with particular values of such observables can be written. This is done by evaluating the probabilities $P(\{S\})$ for various configurations $\{S\}$ of the system.

$$\langle G \rangle = \sum_{\{S\}} G(\{S\})P(\{S\}) \tag{5.3}$$

There are various ways in which this formalism arises. One such is the "information theory" approach. From simple considerations, a function (which can be identified with entropy) is constructed to define "missing information". This is maximised, subject to whatever constraints are imposed by our knowledge of a system.

Let the states of the system be labelled $0, 1, 2, \ldots, R$. The "missing information" function is

$$S = -k \sum_{l=0}^{R} p_l \ln(p_l) \tag{5.4}$$

We assume that we know the expectation values of, for example, two observables, X and Y. (The precise number of such observables is not significant.) This means that X and Y have been measured. We assume that the measured values are the expectation values for the system — a hypothesis that may or may not be correct. Then we have constraints

$$\begin{aligned} \sum_l p_l &= 1 \\ \sum_l X_l p_l &= \langle X \rangle \\ \sum_l Y_l p_l &= \langle Y \rangle \end{aligned} \tag{5.5}$$

It is straightforward, by maximising (5.4) subject to (5.5) to show that (absorbing k into λ)

$$p_l = \frac{e^{-\lambda_Y Y_l + \lambda_X X_l}}{\sum_l e^{-\lambda_Y Y_l + \lambda_X X_l}} \tag{5.6}$$

with λ_Y, λ_X Lagrange multipliers whose values are fixed by

$$\begin{aligned} \langle X \rangle &= -\frac{\partial}{\partial \lambda_X} \ln(Z) \\ \langle Y \rangle &= -\frac{\partial}{\partial \lambda_Y} \ln(Z) \end{aligned} \tag{5.7}$$

where Z is again the **partition function**, the denominator of (5.6).

This gives us a way to determine the p_l, if we can state some constraints on observable properties of the system. All that we need do is to invert (5.7) to determine the λ_X, λ_Y and then substitute them into (5.6). We will discuss this later in the context of the cooccurrence matrices. First, we discuss some simple statistical mechanics models in order to clarify the concepts involved.

5.1.2 Example: The Potts Model

This is a model spin system. The spins occupy the sites of a rigid lattice. At each site, the spin may assume one of P discrete values (in some models the values may be continuous). The energy of the system is defined as some function of these values. Grey level images can be described by Potts models in 2D. The P states at each site correspond to P possible grey levels for each pixel. For a good general survey of this model, see [143]. For a briefer treatment [4] is good.

The (standard) Potts model has the Hamiltonian (energy)

$$H = -\epsilon \sum_{ij} \delta(S_i, S_j) \tag{5.8}$$

where S_i and S_j are spins and the sum is over nearest neighbour sites. ϵ is a constant; if it is negative, the model is antiferromagnetic and if it is positive the model is ferromagnetic, because the lowest energy configuration is one in which all states are the same — a blank image.

A special case of this model is when there are two spin states; labelling them ± 1 rather than $0, 1$ this reduces to the (zero field) Ising model. This is one of the special cases of the Potts model which has been solved exactly in two dimensions. This 2D model is analogous to a binary image. Other models are discussed in [4]. Extensive numerical work has been done on such models.

5.1.3 Example: The General Ising Model

In this simple model of a magnet, N spins can point either "up" ($\uparrow$) or "down" ($\downarrow$). A state of the system is specified by the direction of all the spins, and the Hamiltonian is a function of them.

$$H(\{S\}) = E_0(\{S\}) - F \sum_i S_i \tag{5.9}$$

with F the so called external field. In zero external field, the energy is invariant under spin reversal, so that E_0 is an even function[4, p. 15]. The partition function is

$$Z_N(F,T) = \sum_{\{S\}} e^{-\left[E_0(\{S\}) - F\sum_{\{S\}} S_i\right]} \tag{5.10}$$

where factors of kT have been absorbed into the other parameters.

A particular case of the Ising model, which we will study, is the 1D model, where we consider a chain of N spins like $\uparrow\uparrow\downarrow\uparrow\downarrow\uparrow\uparrow\downarrow \ldots \downarrow$ which have values ± 1. The Hamiltonian is

$$H = \sum_i K S_i S_{i+1} + F S_i \tag{5.11}$$

There are only nearest neighbour interactions in this model. (Some generalisations add second, third, etc neighbour interactions.)

The model has been solved exactly in the 1D case, and also in the 2D zero field case.

5.1.4 Cooccurrence Matrices

We now consider how the expectation values of the cooccurrence matrix elements are defined in this framework. The expectation values of the elements of the cooccurrence matrix (2.12) are given by

$$\langle P_{ab}(\mathbf{x}')\rangle = \frac{1}{Z}\sum_{\{S\}}\left(\frac{1}{N_G}\sum_{\mathbf{x}}\delta(S_{\mathbf{x}},a)\delta(S_{\mathbf{x}+\mathbf{x}'},b)\right)P(\{S\}) \tag{5.12}$$

where $P(\{S\})$ is the probability associated with a particular configuration of the system; we will refer to such a particular configuration as an **image**. The sites are labelled by $\mathbf{x}$, the $\mathbf{x}'$ are displacements in the image and there are N_G pixel levels $0, 1, \ldots, N_G - 1$ in our images which are analogous to the discrete spins in a model such as the Ising or Potts model.

It is important to be clear what the cooccurrence matrix tells us. We are given the results of particular **measurements** made on an image (or images). In order to establish the underlying process behind an image — perhaps to (re)generate it — we need a **model** of this process. This model will contain variable parameters which we will fix from our observations. For instance, we may regard the image as representing a spin system — such as a Potts model — which can be realised by doing a Monte-Carlo simulation, given a Hamiltonian. By using the information which we are given (the cooccurrence matrix) we can obtain this Hamiltonian and use it to generate images. The extent to which this process succeeds will depend upon the correctness of the model and the quality and quantity of the data available.

Each image gives a particular cooccurrence matrix. Given the ensemble of all possible images (a natural concept for images generated by spin systems but a less obvious one for "natural" images, where we may have only one example) the cooccurrence matrices will have some distribution of their own. We label the cooccurrence matrix for image m P^m_{ab} and assign the matrices probabilities P_m so that we can put

$$\langle P_{ab} \rangle = \sum_m P^m_{ab} P_m \tag{5.13}$$

In the network training of Chapters 3 and 4, we use exclusively texture measures derived from cooccurrence matrices $P_{ab}(x, 0)$ defined along the x axis. This is an arbitrary choice — we could have used any direction, or matrices defined for several directions. We justify our choice by the fact that the graphs of measures plotted against x' and y' are close to being rotationally symmetric. (Compare the graphs of measures against x' and y' in Figures 4.1 - 4.5 with the graphs for the correlated random fields in Chapter 3.) However, it should be apparent that for the purposes of regenerating a 2D image, one matrix is not adequate. One could

correctly regenerate[1] **strips** in the direction of that matrix, but perpendicular to these strips the pixels would be random. Thus, at least two matrices are required.

Given these matrices, one could use the method of §5.1.1 to fix a Hamiltonian which would give the probabilities associated with different configurations or images. (We would use the known values of $P_{ab}(\mathbf{x}_1)$ and $P_{ab}(\mathbf{x}_2)$ as the $2N_G^2$ constraints.) A Monte Carlo simulation[7] would then generate the ensemble of images.

We now consider the expected form of the cooccurrence matrix for the two simple models which were described above, the Potts model and the 1D Ising model.

5.1.5 Potts Model Cooccurrence Matrices

Because the interactions in (5.8) concern only neighbouring pixels (or sites, or spins) it is possible to deduce from them the general form of the mean nearest neighbour cooccurrence matrix. First, note that (5.8) has the symmetry of the underlying lattice — a cubic symmetry. We assume that the cooccurrence matrices will have this symmetry and so that the expectation values of their elements will be the same for matrices defined along the axes. This assumption may not always be true.

Systems such as those discussed here often have a **critical temperature** T_c. Above this temperature, the system is disordered. As the temperature falls, local correlations increase until at T_c they exist on all length scales. One can follow this process by calculating the value of some **order parameter**, a function of the configuration such as **magnetisation**

$$M = \langle S_i \rangle = \frac{1}{N} \sum_{i=0}^{N} S_i \tag{5.14}$$

[1]This assumes some sort of stochastic process for the regeneration.

which is zero above T_c and assumes some finite value below it. Since in the ordered system the magnetisation can take on one of several values which break the spin reversal symmetry, we say that spontaneous symmetry breaking takes place in the model. The magnetisation is equal to the mean greyscale level μ of an image.

$T > T_c$ For systems that possess a critical temperature $T_c > 0$ (for example the 2D Potts model with zero field) there is a "disordered" phase above this critical temperature, with no spontaneous magnetisation.

The Potts Hamiltonian (5.8) has one coupling $(-\epsilon)$ for pairs of spins that are the same and another (0) for pairs of spins that differ. We expect that the latter case, represented by the off diagonal elements in P_{ab}, will occur with the same frequency while the former case, represented by the on diagonal elements, will occur with some other frequency.

The nearest neighbour cooccurrence matrix assumes the following expected form (using polar coordinates)

$$\langle P(1,\Theta)\rangle = \begin{pmatrix} \alpha & \beta & \beta & \cdots & \beta \\ \beta & \alpha & \beta & \cdots & \beta \\ \beta & \beta & \alpha & & \\ \vdots & \vdots & \ddots & \ddots & \\ \beta & \beta & & & \alpha \end{pmatrix} \tag{5.15}$$

representing an image which has a random background (the off diagonal elements are the same, so the probability of any pair of pixels is equal) with some correlation (or anticorrelation) between pixels which have the same value (the on diagonal elements).

Note that the Potts Hamiltonian contains interactions between all nearest neighbours, not just those neighbours whose values are used to determine the cooccurrence matrix elements in some direction.

We may fix β in terms of α by using the normalisation condition

$$\sum_{ab} P_{ab} = 1$$

giving

$$\beta = \frac{1 - N_G\alpha}{N_G(N_G - 1)} \tag{5.16}$$

so that exact values for all of the measures defined in (2.15 - 2.19) could be written down in terms of α. For instance,

$$E = \frac{N_G^3\alpha^2 - 2N_G\alpha + 1}{N_G(N_G - 1)} \tag{5.17}$$

$\mathbf{T < T_c}$ Now consider a system which has undergone a transition to a phase with long range order, and a definite non-zero magnetisation. For some systems, no such phase is available for any temperature $T > 0$ (for example, the 1D Potts model with no external field). We will assume that such a phase is available. Further, in order to understand the behaviour of our system and to construct the cooccurrence matrix we will assume that the symmetry has been broken by the addition of a field term to the Hamiltonian:

$$H \rightarrow -\epsilon \sum_{ij} \left(\delta_{S_iS_j} + FS_i\right)$$

where F is a field. This external field may be real, or we may use it to select the direction of spontaneous symmetry breaking and then set it to zero. By rewriting the configuration energy in terms of pairs of spins, one can see that the contribution to the Hamiltonian of a pair of spins ab is the same as that of a pair ba. Thus, the cooccurrence matrix is symmetric

$$P_{ab} = P_{ba}$$

(This property is not true in general for cooccurrence matrices derived from arbitrary images.) Without loss of generality we consider the case of ordering along the 0 direction. Then the form for the cooccurrence matrix is

$$\langle P(1,\Theta)\rangle = \begin{pmatrix} \alpha_0 & \beta_1 & \beta_1 & \cdots & \beta_1 \\ \beta_1 & \alpha_1 & \beta_2 & \cdots & \beta_2 \\ \beta_1 & \beta_2 & \alpha_1 & & \\ \vdots & \vdots & & \ddots & \beta_2 \\ \beta_1 & \beta_2 & & \beta_2 & \alpha_1 \end{pmatrix} \tag{5.18}$$

Normalisation would enable us to fix one degree of freedom. In the limit of low temperature, we would get a "blank" image with $\alpha_0 = 1$ and all other elements of the cooccurrence matrix 0.

The Ising model is represented by the special case of the upper left 2×2 matrices from (5.15) and (5.18).

We will now perform exact calculations of the expectation values of the cooccurrence matrix elements for the 1D Ising model, in zero and in finite fields. Drawing upon these, it will be possible to calculate the cooccurrence matrix elements for the 2D Ising case (in zero field) as well.

5.2 Cooccurence Matrix of 1D Ising Textures

The 1D Ising model (5.11) is especially suitable to be treated via the cooccurrence matrix since the model explicitly concerns a chain of spins, or pixels, as does the cooccurrence matrix; also, the nearest neighbour interactions of the usual Ising model are wholly defined if one can state the nearest neighbour cooccurrence matrix.

5.2.1 Model

With a finite field, the Hamiltonian (5.11) becomes

$$-\sum_{i=1}^{N-1}(JS_iS_{i+1} + FS_i) \tag{5.19}$$

where F is an applied external field. Put

$$K = -\frac{J}{kT}$$

and

$$L = \frac{F}{kT}$$

Then from the Boltzman distribution (5.1) already defined, the probability of a state described by the set of spins $\{S\}$ is

$$P(\{S\}) = \frac{1}{Z}\exp(K\sum_{i=0}^{N} S_iS_{i+1} + LS_i) \tag{5.20}$$

Zero Field

The partition function in the zero field case is (from [4])

$$Z = 2^N \cosh^N K \tag{5.21}$$

There are many ways to derive this result. One[130, p. 132] is in terms of a transfer matrix T which carries the spin at a site i to that at $i+1$. The transfer matrix is

$$\begin{bmatrix} e^K & e^{-K} \\ e^{-K} & e^K \end{bmatrix}$$

Then for periodic boundary conditions

$$\begin{aligned} Z &= \sum_{\{S_i=\pm 1\}} \prod_{i=1}^{N} e^{KS_{i+1}S_i} \\ &= Tr(\mathbf{T}^N) \end{aligned} \tag{5.22}$$

where Tr is the trace operator. The transfer matrix has eigenvalues λ_0 and λ_1 given by $e^K \pm e^{-K}$. The corresponding eigenvectors are

$$|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}+1\\+1\end{bmatrix} = \frac{1}{\sqrt{2}}(|+\rangle + |-\rangle)$$

and

$$|1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}+1\\-1\end{bmatrix} = \frac{1}{\sqrt{2}}(|+\rangle - |-\rangle)$$

where $|+\rangle$ and $|-\rangle$ are the eigenvectors of the Pauli matrix σ_3

$$\sigma_3 = \begin{bmatrix}1 & 0\\0 & -1\end{bmatrix} \tag{5.23}$$

The partition function can therefore be written $\lambda_0^N + \lambda_1^N$, where $\lambda_0 > \lambda_1$. For large N λ_0 determines the partition function, yielding (5.21).

Finite Field

When there is a finite field, the partition function is given by

$$Z_N = \lambda_0^N + \lambda_1^N \tag{5.24}$$

with λ_0 and λ_1 the eigenvalues of the transfer matrix

$$\begin{bmatrix}e^{K+H} & e^{-K}\\e^{-K} & e^{K-H}\end{bmatrix}$$

which are given by

$$e^K\left(\frac{e^H + e^{-H}}{2}\right) \pm \sqrt{e^{-2K} - e^{2K} + e^{2K}\left(\frac{e^H + e^{-H}}{2}\right)^2} \tag{5.25}$$

Again, for large N, the properties of the partition function are dominated by λ_0, the larger of the two eigenvalues. All of this is discussed in [4] and [130]. We proceed to derive the expectation values of the cooccurrence matrix elements in both the zero and the finite field cases.

5.2.2 Zero Field Case.

Since the partition function Z in (5.21) has no singularity, the 1D Ising model does not have a phase transition at any finite temperature (with Stanley [130], we may say that it has a transition at zero temperature). Thus we can write the cooccurrence matrix in the form

$$\frac{1}{N}\begin{pmatrix} A & \frac{N}{2}-A \\ \frac{N}{2}-A & A \end{pmatrix}$$

$A = N\alpha$ and $\frac{N}{2} - A = N\beta$ represent the numbers of pairs of adjacent pixels (spins, sites) found respectively with the same and with different values. The nearest neighbour configurations $\downarrow\uparrow$ and $\uparrow\downarrow$ make the same contribution to the Hamiltonian and thus are equally likely; the same is true for $\uparrow\uparrow$ and $\downarrow\downarrow$. We consider two methods for derivation of the cooccurrence matrix elements; transfer matrices, and a simpler method, based on known correlation functions.

Use of Transfer Matrix Method

We construct a unitary matrix $\mathbf{R}$ from the eigenvectors of $\mathbf{T}$, which diagonalises $\mathbf{T}$.

$$\mathbf{R} = \frac{1}{\sqrt{2}}\begin{pmatrix} +1 & +1 \\ +1 & -1 \end{pmatrix}$$

so that

$$\mathbf{R}^{\dagger}\mathbf{T}\mathbf{R} = \begin{bmatrix} \lambda_0 & 0 \\ 0 & \lambda_1 \end{bmatrix} \tag{5.26}$$

and

$$\left(\mathbf{R}^{\dagger}\mathbf{T}\mathbf{R}\right)^m = \begin{bmatrix} \lambda_0^m & 0 \\ 0 & \lambda_1^m \end{bmatrix} \tag{5.27}$$

We wish to calculate the expectation values of the cooccurrence matrix elements for the 1D Ising model in a zero external field. These are given by

$$\langle P(k')_{ab} \rangle = \frac{1}{Z} \sum_{\{S\}} \frac{1}{N} \sum_{k} \delta(S_k - a)\delta(S_{k+k'} - b) \times e^{K \sum_{l=0}^{N-1} S_l S_{l+1}} \tag{5.28}$$

with Z given by (5.21).

By using the method of transfer matrices, one may express this configuration sum as the trace of a matrix[4]. Since the spins are ± 1, it is possible to write

$$\delta(S_k - a) = \frac{1}{4}(S_k + a)^2$$

We wish to express the configuration sum (5.28) as the trace of a matrix. If we were calculating the correlation function $\langle \sigma_i \sigma_j \rangle$, the Pauli matrix (5.23) would be used to represent the two spins whose correlation was being determined. Instead, to calculate the cooccurrence matrix elements we use

$$\mathbf{D}(a) = \frac{1}{4} \begin{bmatrix} (a-1)^2 & 0 \\ 0 & (a+1)^2 \end{bmatrix} \tag{5.29}$$

to represent the delta functions. We are determining the cooccurrence matrix element P_{ab}. The matrix whose trace we wish to evaluate, then, is

$$\mathbf{D}(a)\mathbf{T}^{(j-i)}\mathbf{D}(b)\mathbf{T}^{(N-(j-i))}$$

We actually calculate

$$(\mathbf{R}^\dagger \mathbf{D}(a)\mathbf{R})(\mathbf{R}^\dagger \mathbf{T}^{(j-i)}\mathbf{R})(\mathbf{R}^\dagger \mathbf{D}(b)\mathbf{R})(\mathbf{R}^\dagger \mathbf{T}^{(N-(j-i))}\mathbf{R})$$

where the $(j-i)$th power of the transfer matrix is used to carry the system between two sites. Eventually, we let $N \to \infty$. With the normalisation $\frac{1}{Z_N}$, this will give the required cooccurrence matrix elements. Now, using $a = \pm 1$ we can

show that

$$\mathbf{R}^{\dagger}\mathbf{D}(a)\mathbf{R} = \frac{1}{2}\begin{bmatrix} 1 & -a \\ -a & 1 \end{bmatrix}$$

We calculate, using this result and (5.27)

$$\langle P_{ab}(j-i)\rangle = \frac{1}{4}\frac{1}{\lambda_0^N}\left[\lambda_0^N\left(1+ab\left(\frac{\lambda_1}{\lambda_0}\right)^{(j-i)}\right)+\lambda_1^N\left(1+ab\left(\frac{\lambda_0}{\lambda_1}\right)^{(j-i)}\right)\right] \quad (5.30)$$

Given $\lambda_0 > \lambda_1$, as $N \rightarrow \infty$, we are left with

$$\langle P_{ab}(j-i)\rangle = \frac{1}{4}\left[1+ab\left(\frac{\lambda_1}{\lambda_0}\right)^{(j-i)}\right] \quad (5.31)$$

In (5.31), it is clear that the terms on the leading diagonal are equal to one another, as are the off diagonal terms. This is to be expected for the 1D Ising model in zero external field; there cannot be any breaking of the $\uparrow\uparrow$ and $\downarrow\downarrow$ symmetry.

Use of Correlation Function Method

There is an alternative and more direct way to obtain (5.31). The correlation function is already known for the Ising model, but it can also be written in terms of our conjectured form for the cooccurrence matrix. We have

$$P(j-i) = \frac{1}{N}\begin{bmatrix} A & \frac{N}{2}-A \\ \frac{N}{2}-A & A \end{bmatrix} = \begin{bmatrix} \alpha & \frac{1}{2}-\alpha \\ \frac{1}{2}-\alpha & \alpha \end{bmatrix} \quad (5.32)$$

Then the two expressions can be set equal, and the free parameter α eliminated between them. From (5.32), the correlation is

$$\kappa(j-i) = \sum_{ab} abP_{ab} = 4\alpha - 1 \quad (5.33)$$

But the correlation function is given[130, 4] by

$$\begin{aligned} \gamma(x) &= \langle S_i S_{i+x}\rangle \\ &= \left(\frac{\lambda_1}{\lambda_0}\right)^{j-i} \end{aligned} \quad (5.34)$$

for a displacement $|j-i| = x$. (This result is obtained by transfer matrices or other methods). Equating $\gamma(x)$ (5.33) and $\kappa(x)$ (5.34) and solving for α we find that

$$\alpha = \frac{1}{4}\left[1 + \left(\frac{\lambda_1}{\lambda_0}\right)^{j-i}\right]$$

which we substitute in (5.32), giving the same results for the cooccurrence matrix as those found by use of the transfer matrix. Although this method seems simpler, to prove (5.34) requires, for example, use of the transfer matrix formalism and so the two methods are equivalent.

We now relate our derived values for the cooccurrence matrix elements to some other parameters, beginning with correlation length.

Correlation Length and Long Distance Behaviour

Since we can write the cooccurrence matrix in terms of correlation functions, we can introduce a correlation length. Define

$$\begin{aligned} g_{ij} &= \langle S_i S_j \rangle - \langle S_i \rangle \langle S_j \rangle \\ &= \gamma(x) - \mu^2 \end{aligned} \tag{5.35}$$

Translational invariance means that $\langle S_i \rangle$ is the same for all sites (for the zero field 1D Ising model it is zero) and that g_{ij} depends only on the separation between two sites, $x = |j-i|$. We write $g_{ij} = g(x)$.

Far from T_c (which is zero in this case) it is expected [4] that the correlation function $g(x)$ obeys the following law for large x:

$$g(x) \approx x^{-\tau} e^{-\frac{x}{\xi}} \tag{5.36}$$

where ξ is the correlation length, and τ is some constant. Now, since the cooccurrence matrix explicitly contains the correlation function γ we can rewrite the

cooccurrence matrix (5.31) as

$$\langle P_{ab}(j-i)\rangle = \frac{1}{4}\left[1+\omega ab\left(e^{-\frac{(j-i)}{\xi}}\right)\right] \tag{5.37}$$

where ω is some constant. Clearly, for $|j-i| \gg \xi$, $P_{ab} \to \frac{1}{4}$. For a given value of T (or K) the correlation length can be calculated; for $T = 0$, it is

$$\left[\log\left(\frac{\lambda_0}{\lambda_1}\right)\right]^{-1}$$

From (5.37) it is clear that for large $|j-i|$, $P_{ab} \to \frac{1}{4} = \frac{1}{N_G^2}$ in agreement with the forms in Chapter 2 and with Table 4.1. We can proceed to calculate the texture measures in this limit, and show that they also agree with Chapter 2.

Texture Measures

Given the calculated expectation values of the cooccurrence matrix elements, it is possible to calculate expectation values for the texture measures themselves. The following results are general; in the specific limit $|j-i| \to \infty$ they agree with the limit discussed in Chapter 2, except for the differences due to the pixel values being ± 1 rather than $0, 1$ as is assumed in Chapter 2. In Table 4.1 this agreement may be seen. The following results are exact except for Entropy, which is an approximation for $|j-i| \gg 1$.

Energy

$$E(j-i) = \frac{1}{4}\left[1+\left(\frac{\lambda_1}{\lambda_0}\right)^{(j-i)^2}\right] \tag{5.38}$$

Entropy In deriving this result, it was necessary to use the approximation

$$\log\left(1+ab\left(\frac{\lambda_1}{\lambda_0}\right)^{(j-i)}\right) \approx ab\left(\frac{\lambda_1}{\lambda_0}\right)^{(j-i)}$$

which is valid for large $|j-i|$ since $ab \approx 1$ and $\lambda_1 < \lambda_0$

$$H(j-i) = \log 4 - \left(\frac{\lambda_1}{\lambda_0}\right)^{(j-i)^2} \tag{5.39}$$

Correlation The correlation texture measure is identical to the correlation function $g(|j-i|)$ discussed above,

$$C(j-i) = \left(\frac{\lambda_1}{\lambda_0}\right)^{(j-i)} \tag{5.40}$$

Local Homogeneity

$$L(|j-i|) = \frac{1}{10}\left[6 + 4\left(\frac{\lambda_1}{\lambda_0}\right)^{j-i}\right] \tag{5.41}$$

Inertia

$$I(j-i) = 2\left[1 - \left(\frac{\lambda_1}{\lambda_0}\right)^{j-i}\right] \tag{5.42}$$

All of these measures tend to a limit at least as rapidly as the correlation; they are particular cases of $N_G = 2$. If one were to perform Ising simulations for various different values of K, one would find the same limiting behaviour — compare with the graphs showing the texture measures against x' for the rocks (4.1 - 4.5) where the limiting behaviours are different. However if $H \neq 0$ then the limiting cases of the texture measures would not be identical. In such a case, the first order probability of the pixel values would not be the same and would depend on H.

Comparing Figures 5.1 - 5.5 with the graphs showing texture measures for the limestones, Figures 4.1 - 4.5, we observe that the overall behaviour is qualitatively the same for the two cases. This suggests that the dependence of the texture measures on the correlation function is similar in nature for the limestones and

the Ising textures even though in the latter case we are not able to calculate it explicitly.

We now turn to the finite field case and repeat our earlier calculation of the cooccurrence matrix elements. The same two methods are used.

5.2.3 Finite Field Case.

The above calculations can be repeated for the case of a non-zero external field.

Use of Transfer Matrices

The working in this case is similar to that for the zero field case. The transfer matrix is now

$$\begin{bmatrix} e^{K+H} & e^{-K} \\ e^{-K} & e^{K+H} \end{bmatrix}$$

And it has eigenvalues given by (5.25).

With Baxter [4], we put

$$\cot(2\phi) = e^{2K}\sinh(H)$$

and the eigenvectors of the transfer matrix are now

$$\begin{bmatrix} +\cos(\phi) \\ +\sin(\phi) \end{bmatrix}$$

and

$$\begin{bmatrix} +\sin(\phi) \\ -\cos(\phi) \end{bmatrix}$$

So that

$$\mathbf{R} = \begin{pmatrix} +\cos(\phi) & +\sin(\phi) \\ +\sin(\phi) & -\cos(\phi) \end{pmatrix}$$

and (as before, but with a different $\mathbf{R}$ and $\mathbf{T}$)

$$\mathbf{R}^{\dagger}\mathbf{T}\mathbf{R} = \begin{bmatrix} \lambda_0 & 0 \\ 0 & \lambda_1 \end{bmatrix} \tag{5.43}$$

Further,

$$\mathbf{R}^{\dagger}\mathbf{D}(a)\mathbf{R} = \frac{1}{2}\begin{bmatrix} 1 - a\cos 2\phi & -a\sin 2\phi \\ -a\sin 2\phi & 1 + a\cos 2\phi \end{bmatrix}$$

We calculate

$$\begin{aligned} \langle P_{ab}(j-i)\rangle &= \frac{1}{4}\frac{1}{\lambda_0^N}Tr \\ &\begin{bmatrix} 1 - a\cos 2\phi & -a\sin 2\phi \\ -a\sin 2\phi & 1 + a\cos 2\phi \end{bmatrix} \\ &\times \begin{bmatrix} \lambda_0^{(j-i)} & 0 \\ 0 & \lambda_1^{(j-i)} \end{bmatrix} \\ &\times \begin{bmatrix} 1 - b\cos 2\phi & -b\sin 2\phi \\ -b\sin 2\phi & 1 + b\cos 2\phi \end{bmatrix} \\ &\times \begin{bmatrix} \lambda_0^{(N-(j-i))} & 0 \\ 0 & \lambda_1^{(N-(j-i))} \end{bmatrix} \end{aligned} \tag{5.44}$$

In the limit of $N \to \infty$ we obtain

$$\langle P_{ab}(j-i)\rangle = \frac{1}{4}\left[(1 + a\cos(2\phi))(1 + b\cos(2\phi)) + (ab\sin^2 2\phi)\left(\frac{\lambda_1}{\lambda_0}\right)^{(j-i)}\right] \tag{5.45}$$

The symmetry of the elements down the leading diagonal has been broken by the imposition of the field, even though there is still no non zero critical point.

In the limit $H \to 0$, this matrix reduces to the zero field case. As $|j-i| \to \infty$,

$$P_{ab} \to \frac{1}{4}\left[(1 + a\cos(2\phi))(1 + b\cos(2\phi))\right] \neq \frac{1}{4}$$

so that the measures do not assume their "average" values as in the zero field case above, or the large distance limit of Chapter 2. This is because the pixel

values are not equally probable in the finite field Ising model. (Compare with Chapter 4 where this is discussed in the context of rock images.)

This result clearly agrees with the zero field result as in the limit, $\langle S_i \rangle \to 0$. However, for finite fields the symmetry on the leading diagonal is broken, as we note above.

Texture Measures

In the same way as we did for the zero field case, we can use our calculated values of $\langle P_{ab}(j-i) \rangle$ to determine expectation values for the texture measures. The expressions are more complicated in general, and we do not give them here, but the working is similar. Note, however, that as for the zero field case, the correlation texture measure is identical to $g(|j-i|)$ (required by the sum rule mentioned above).

Use of Correlation Function

As in the zero field case, we may obtain (5.45) by direct use of the known correlation function for the model[130, 4].

Using our conjectured form of the cooccurrence matrix,

$$\begin{bmatrix} \alpha+\delta & \frac{1}{2}-\alpha \\ \frac{1}{2}-\alpha & \alpha-\delta \end{bmatrix} \tag{5.46}$$

where as before we have used normalisation to remove β, we now need to fix α and δ. In this case, we turn to the magnetisation[2]. It is known[4], for the finite field Ising model, that

$$M = \cos 2\phi$$

but also, from (5.46) by considering the contribution to the magnetisation of each

[2]In the zero field case, magnetisation was zero so this meant that δ was also zero.

pair of spins (or by calculating μ which is equal to the magnetisation)

$$M = -2\delta \tag{5.47}$$

So we have fixed δ. We now proceed to use the correlation function given in [4]

$$\langle S_i S_j \rangle = \cos^2 2\phi + \left(\frac{\lambda_1}{\lambda_0}\right)^{j-i} \sin^2 2\phi \tag{5.48}$$

to fix α, giving the same results as the previous method.

Correlation Length From the cooccurrence matrix, it is easy to see that $\sum_{ab} abP_{ab}$ gives the correlation, as it does in the zero field case. Again, we can write the cooccurrence matrix in terms of correlation functions and correlation lengths. Using (5.35) as before, but now with $\langle S_i \rangle = \cos(2\phi)$ we will write $\langle S_i \rangle = \mu$ and using (5.48) we have

$$\langle P_{ab} \rangle = \frac{1}{4}\left[(1 + a\mu)(1 + b\mu) + ab\left(e^{-\frac{(j-i)}{\xi}}\right)\right] \tag{5.49}$$

For $|j - i| \gg \xi$, the exponential term vanishes, giving

$$P_{ab} = \frac{1}{4}\begin{bmatrix} (1+\mu)^2 & (1-\mu^2) \\ (1-\mu^2) & (1-\mu)^2 \end{bmatrix}$$

in agreement with (5.45).

5.3 2D Ising Model

We can repeat the correlation function method used above for the 2D Ising model in zero field, since its correlation function is known (in fact, we could repeat the method for any model defined on a discrete lattice whose two point correlation funtion is known).

In this case, the form of the expected cooccurrence matrix in a particular direction is (as above)

$$\begin{bmatrix} \alpha+\delta & \beta \\ \beta & \alpha-\delta \end{bmatrix}$$

Since the matrix is normalised, $\beta = \frac{1}{2} - \alpha$. We assume that the 2D Ising model has identical couplings in the two orthogonal directions. Then the average properties of the ensemble of configurations will be the same in these two directions, and so the expectation values of the cooccurrence matrix will also be the same.

Now, an expression is known for the magnetisation for the zero field 2D Ising model[4].

$$M_0 = (1 - k^2)^{\frac{1}{8}} \tag{5.50}$$

where

$$k = \left(\sinh^2 2K\right)^{-1}$$

with K the coupling constant. We also know that $M_0 = \mu$ as before; μ is defined in Chapter 2. We find that

$$\mu = -2\delta$$

so that

$$\delta = -\frac{1}{2}\mu \tag{5.51}$$

Compare (5.47). At this point, the cooccurrence matrix is quite general; no restriction has been placed on (x', y') since any cooccurrence matrix will give the same value for the magnetisation. We will restrict the type of matrix considered to those developed parallel to the x and y axes. The form we have for the cooccurrence matrix is

$$\begin{bmatrix} \alpha+\frac{1}{2}\mu & \frac{1}{2}-\alpha \\ \frac{1}{2}-\alpha & \alpha-\frac{1}{2}\mu \end{bmatrix}$$

The cumulant is defined as

$$\begin{aligned} & \langle S_i S_j \rangle - \langle S_i \rangle \langle S_j \rangle \\ = \quad & \langle S_i S_j \rangle - \mu^2 \end{aligned} \tag{5.52}$$

since we have translational invariance (we are assuming $|j - i| = x$). Then

$$\langle S_i S_j \rangle = 4\alpha - 1$$

so that the cumulant is

$$4\alpha - 1 - \mu^2 \tag{5.53}$$

But we know an expression for this cumulant at large distance; it is given by Baxter[4, p. 117] and is

$$\approx \left(\frac{\Lambda_2}{\Lambda_{MAX}}\right)^x \tag{5.54}$$

where x is the distance between two sites on the lattice (the displacement used to calculate the cooccurrence matrix is $\overrightarrow{PQ}$ and x is the distance between P and Q.) The transfer matrix is a more complicated object than hitherto and it has eigenvalues

$$\Gamma_{MAX} > \Gamma_1 > \Gamma_2 \ldots$$

Equating (5.53) and (5.54) we find that

$$\alpha = \frac{1}{4}\left(1 + \mu^2 + \left(\frac{\Lambda_2}{\Lambda_{MAX}}\right)^x\right) \tag{5.55}$$

and substituting for α, we find that

$$P_{ab} = \left[(1 + a\mu)(1 + b\mu) + ab\left(\frac{\Lambda_2}{\Lambda_{MAX}}\right)^x\right] \tag{5.56}$$

These results [(5.31), (5.45) and (5.56)] agree with the general forms given by Carstensen[20, p. 33] for the grey level cooccurrence matrices of binary images.

They were derived in the same general way, so it is not surprising that they agree in their form. The detailed expressions for the correlation functions themselves are different, and that for the 2D case is approximate. However, the finite field expressions reduce to the zero field case when magnetisation is zero.

In our derivation of the 2D expression, we did not make any assumptions about the temperature, so our expression is valid both above and below T_c. It has an explicit dependence on the magnetisation, which is zero above T_c and which takes some finite value below T_c.

Since the 2D Ising model in zero field has not been solved, we have not calculated an explicit expression for the cooccurrence matrix elements in this case.

5.4 Image Regeneration.

It is interesting to ask whether, given one cooccurrence matrix (or more) it is possible to (re)generate an image with the correct statistics. If this can be done, does the image resemble the original one from which the matrix was derived? Does it suggest that the cooccurrence matrix has captured some important information about the image, so that it can be regarded as a reliable measure for texture recognition? Do the derived texture measures retain such information?

Many areas of research use textural information and image regeneration, and the ability to use the cooccurrence matrices, for instance, might be useful to generate textures with controlled properties.

A brief survey of the available literature shows that the regeneration of textured images has been extensively addressed. [20] is a good study, which uses Markov random fields to generate textures. Various techniques are employed for generating these fields, and the results are applied to the simulation of geological

structures, in particular to reservoir modelling.

The cooccurrence matrices themselves are used in [40]. An initial, starting configuration is generated with the same first order probabilities as the desired image, and this configuration is then modified by a simulated annealing procedure which exchanges the greyscale values for pairs of sites where this reduces an error measure defined on the difference between the actual and desired P_{ab}. This scheme is also motivated by the need to simulate geological formations, as is the method of Smith and Freeze already discussed[127]. The use of stochastic models in geological work is widespread; it is discussed by Ripley[62, Ch. 12] and by Haldorsen and Damsleth[63].

Neural network methods have also been used in this field. Hwang and Chen[75] use a backpropagation network to estimate the local conditional distributions of textured images, instead of doing this in a Markov random field formalism. These approaches are only a few of many which have been attempted.

The cooccurrence matrix itself contains 1D information. One considers the relations between pairs of pixels, and the parameters relating these pairs define a special direction in the plane. Thus, one might expect that image regeneration for the 1D case is relatively straightforward. For the 2D case, it is not so clear; one requires information about two directions and this is embodied in several cooccurrence matrices which must somehow be combined. A Bayesian framework might be appropriate in such circumstances.

It might be possible to use the cooccurrence matrices to regenerate textures directly, rather than by indirect means such as are used in [40]. We will consider how the cooccurrence matrices might be of use in image regeneration, adopting two points of view; first, a more abstract framework (already touched on above) from statistical mechanics.

5.4.1 Cooccurrence Matrices as Constraints

We have an image function $S^l(x)$ with l denoting a particular image, or **configuration**; if there are N_G pixel values, we have N_G^N possible images, so $l = 0, 1, 2, \ldots, N_G^N$.

Now, (5.13) is a constraint equation like (5.5). Following the same working through as before in Chapter 2, we obtain

$$P_m = \frac{e^{-\sum_{ab} \lambda_{ab} P_{ab}^m}}{\sum_m e^{-\sum_{ab} \lambda_{ab} P_{ab}^m}} \tag{5.57}$$

and

$$\langle P_{ab} \rangle = -\frac{\partial}{\partial \lambda_{ab}} \ln(Z) \tag{5.58}$$

Equations such as (5.58) can be solved numerically. Use of methods due to Swendsen [132, 133] enables the Lagrange multipliers λ_{ab} to be determined. Once the λ_{ab} are known, Monte Carlo techniques[7] can be employed to generate the required distribution of images.

Alternatively, one may look for a correspondence between the cooccurrence matrix, which is assumed known, and the transfer matrix.

5.4.2 Direct Use of Cooccurrence Matrices

The cooccurrence matrix elements are joint probabilities, that is

$$P_{ab} = P(S_x = a, S_{x+x'} = b) \tag{5.59}$$

where the notation $P(A, B)$ denotes the probability of events A and B. If we fix b, then, the P_{ab}, suitably normalised, denote the probabilities of finding various pixel values a.

These properties can be used in a scheme to generate image textures, if we have the cooccurrence matrix.

Such a scheme might be implemented, for example, as a Gibbs sampler [20, p. 128]. In such an algorithm, one chooses a starting configuration. A pixel is then chosen, and its value replaced by a value taken from the conditional distribution given the values of its neighbours (it is this distribution which we would calculate from the P_{ab}.) This procedure is repeated until some stop condition is met.

The transfer matrix elements are conditional probabilities, that is,

$$T_{ab} = P(S_i = a | S_{i+1} = b)$$

while the elements of the cooccurrence matrix (nearest neighbour or otherwise) are joint probabilities

$$P(1)_{ab} = P(S_i = a, S_{i+1} = b)$$

Since conditional probabilities and joint probabilities are related by Bayes' Theorem (5.60) we should be able to get one of the matrices from the other. We know that, in general,

$$P(A|B)P(B) = P(B|A)P(A) \tag{5.60}$$

$$P(S_i = \pm 1) = \frac{1}{2}[1 \pm cos(2\phi)] \tag{5.61}$$

in the Ising model, which reduces to a probability of $\frac{1}{2}$ for each direction of spin when there is no field present. We can, then, show that the (nearest neighbour) cooccurrence matrix is derivable from the transfer matrix directly; and since, (in the Ising model) the transfer matrix defines a Markov process that generates configurations, use of the cooccurrence matrix to (re)generate an ensemble of images is equivalent to a Monte Carlo realisation of the Ising model.

5.5 Concluding Remarks

In this chapter we have discussed the cooccurrence matrix elements in the light of statistical mechanics. Use of a statistical mechanics formalism enables us to determine expectation values for the matrix elements when the underlying image is generated by an appropriate model. This means that such models can also be used to create textures whose expected properties are known.

Furthermore, since the cooccurrence matrix contains probabilities, it should be possible to use it to generate textures directly either through the Monte Carlo procedure of §5.4.1 which is general and applies to systems of more than 1D, or through the direct method of §5.4.2 which is more appropriate in the 1D case.

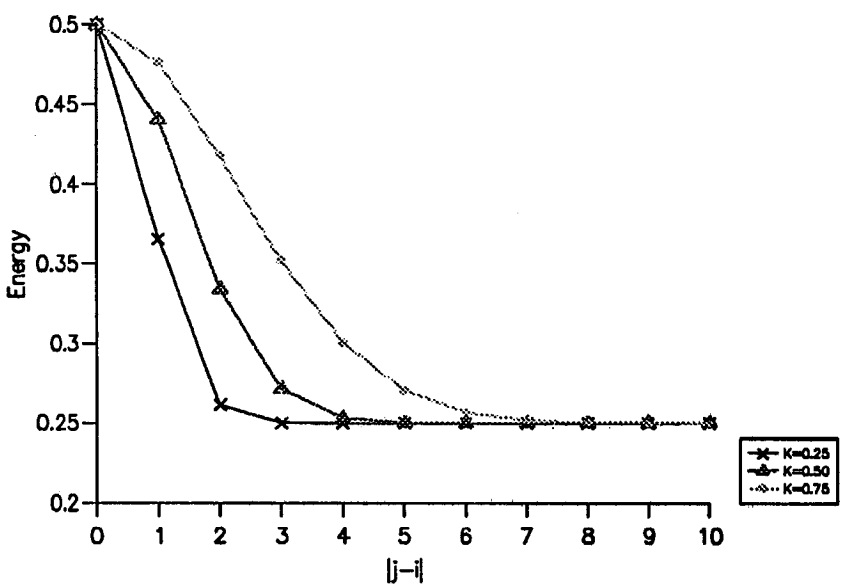

Figure 5.1. Energy measure for texture generated by a zero field Ising model, derived from theoretical result (5.38). The energy is shown for various values of K.

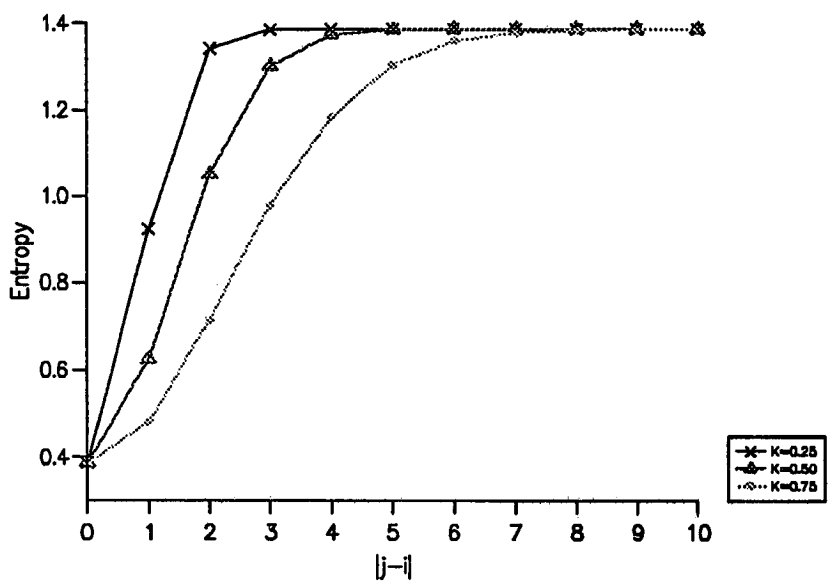

Figure 5.2. Entropy measure for texture generated by a zero field Ising model, derived from theoretical result (5.39). The entropy is shown for various values of K.

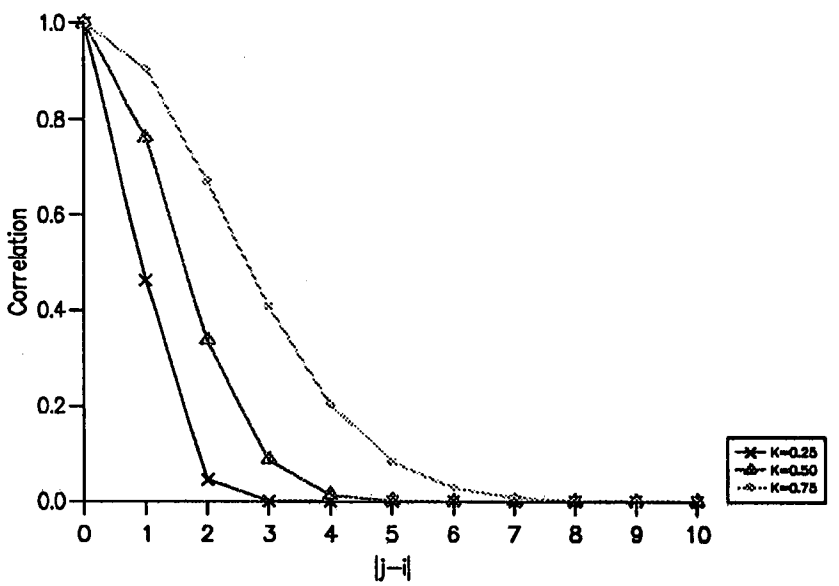

Figure 5.3. Correlation measure for texture generated by a zero field Ising model, derived from theoretical result (5.40). The correlation is shown for various values of K.

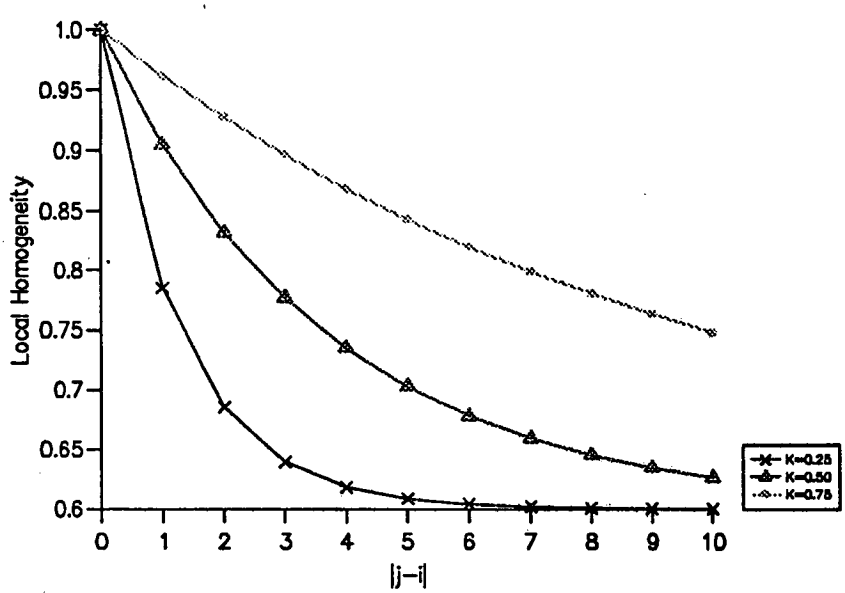

Figure 5.4. Local homogeneity measure for texture generated by a zero field Ising model, derived from theoretical result (5.41). The local homogeneity is shown for various values of K.

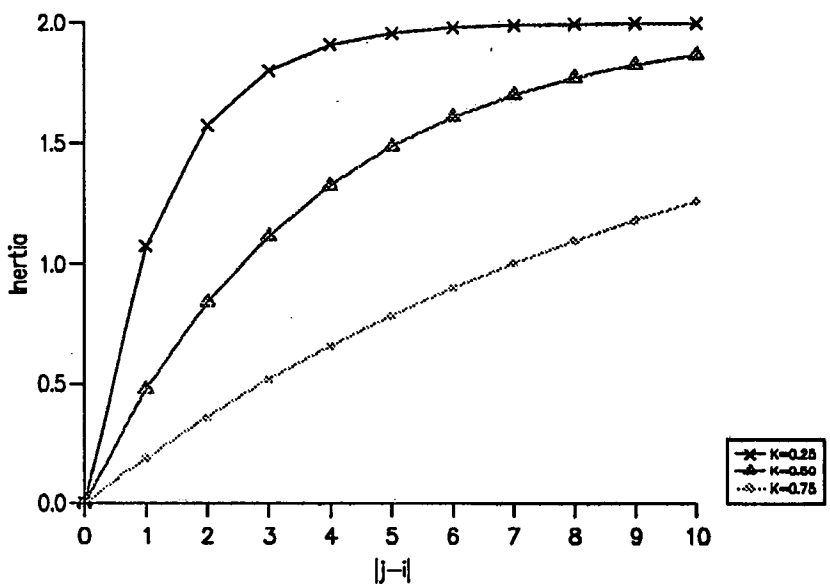

Figure 5.5. Inertia measure for texture generated by a zero field Ising model, derived from theoretical result (5.42). The inertia is shown for various values of K.

Chapter 6

Conclusion and Summary

6.1 Summary

The aim of this thesis was to investigate the problem of lithofacies identification as a specific instance of the general pattern classification problem. Pattern classification raises issues of pattern recognition, of discrimination and of identification and the purpose of investigating lithofacies classification was to examine these issues within the framework of neural networks. In order to do this, techniques from two principal fields were required.

- Neural networks themselves were used for the pattern recognition/ classification proper.
- Cooccurrence matrix based texture measures were used to represent the images which were to be classified.

We have seen (Chapter 1) that research into machine texture recognition is commercially and scientifically important. Extensive research has ben done into the nature and properties of neural networks themselves, and into their use for pattern classification. The work presented here on the first of the two

areas outlined above is not, therefore, greatly original though the results agree pleasingly with what has been done before. But less work has been done on the properties of the cooccurrence matrix based measures (rather than their use, which has been widely studied). Here we believe that we have made an original contribution. By using techniques from statistical mechanics to calculate exact values of the texture measures for models of texture.

The background to the use of neural networks was discussed in Chapter 2. It was shown there that there exists a plausible training algorithm for a general feedforward neural network. This algorithm, the backpropagation algorithm, is the base for much of the network training in the thesis. The justification for using the cooccurrence matrix based texture measures is given in the same chapter, based on the extensive work of Julesz and his co-workers and upon successful results in actual classification applications.

In Chapters 3 and 4 we used the cooccurrence based texture measures to characterise various limestones and sandstones. In order to do this, it was necessary to write software to calculate the values of the measures. (This software is described in Appendix A.) All of this required a great deal of computational work to develop and test procedures for acquiring the image data, and to establish appropriate values for the parameters needed in the calculation of the measures. These parameters describe the size and nature of the net itself — and two different sorts of net were considerd — and the way in which the measures were calculated (the area of image used, the displacement within the image, the degree of coarse graining and the normalisation of the image). They also comprised variable parameters used in learning — update stepsize, noise and so forth. Here, our results showed a dependence on the training parameters, but not enough work was done to identify the best values.

The performance of the network needed to be compared with some other classification method for the same data, or else the results would have been meaningless. So we also classified the rock images (represented by the cooccurrence matrix based texture measures) using the K-nearest neighbours algorithm which is an established classification method (and which has the further virtue of being simple). The results of this work show that, presented with the same input vectors, neural network models can significantly outperform K nearest neighbours methods. Comparisons were also made with classifiaction of the same images by humnas, but here no definite conclusions can be drawn.

Finally, at a slight remove from the work on classification, in Chapter 5, we consider the nature of the cooccurrence matrix based texture measures themselves. This is of interest because of their affinity with the correlation functions known to statistical physics. The work discussed in this chapter is the most original work in the thesis. We calculated texture measures for textures generated by a simple model (the Ising model in a variety of circumstances). For this model, we found general relationships between the cooccurrence matrix element and objects familiar in statistical mechanics, two point correlation functions. There is an extensive literature on the properties of these functions and they play a fundamental part in physics. Thus, we have shown that the cooccurrence matrices are interesting in themselves, not just as measures of textural properties.

The calculated measures agree qualitatively with those which were derived for the rock images, supporting the validity of the computer software used to derive the measures from images. This also suggests that the interrelations between the measures which were found for the Ising case, due to the presence of the two point correlation function in the expression for the cooccurrence matrix elements, are useful to interpret the "natural" rock images.

In short, this thesis

- Shows that the cooccurrence matrix based measures are valid as representations of textures (for a particular class of images).
- Demonstrates that such measures can be used with pattern classification techniques (in particular, neural networks and K-nearest neighbour methods) to classify textures.
- Goes further and investigates the nature of these measures by deriving them from first principles for some textures generated by simple models. The measures are shown, for this texture model, to be simple combinations of a basic measure, the correlation.
- Conjectures that some such relationship holds in general for such texture measures.

Possibilities for further development of this are given below.

6.2 Further Work

The work presented in this thesis by no means exhausts what might be done. These preliminary studies suggest that neural networks have potential as a means of reliably differentiating images of lithofacies types. However, further research is required before this technology can be applied to borehole imagery. First, further work must be done to ensure that the images have been correctly normalised so that differentiation does not proceed solely on the basis of tone. Secondly, it is necessary to define the minimum image size required to capture sufficient textural information to recognise a given lithofacies type. These area scales will be lithofacies specific; the area scale required to recognise a cross laminated, medium grained sand is likely to be smaller than that required to recognise a granule grade conglomerate or a vuggy carbonate.

Neural network techniques will have further applications. The problem of identifying lithofacies, considered simply as textures, is a simplification of what needs to be done in exploration. We may be required to differentiate sequences of lithofacies, for instance; other information, as well as the textural, may be used, and neural networks are ideal for adding this in; one simply uses additional inputs. In this way, well log measurements can supplement the FMS/ FMI images.

The next stage from interpreting sequences of lithofacies is to identify and understand features as an aid to generating a paleogeography; determining the nature of the landscape when the formations were laid down. Such a task could involve coordinating results from more than one well, possibly over a large area, and could probably not be done in real time. It might require the input of other data — such as seismic results — with the borehole results. Again, the nature of neural networks makes this feasible.

We did not mention in the previous section the possibilities of using the cooccurrence matrices to regenerate texture. Textural and image generation is an active field, not least in the field of petroleum engineering. Although work has been done[40] which used cooccurrence matrices as a texture description, we have not found them used for the direct regeneration of texture. Given their nature, this would seem to be straightforward given standard Monta Carlo techniques.

Finally, our method for calculating the expectation values of the cooccurrence matrix elements could be generalised and applied for other models, most obviously to the Potts model.

Appendix A

Parallel Programming

In this Appendix, we describe some of the programs which were written to support the work described in the thesis. These were concerned with the calculation of texture measures. The software which was used to perform neural network simulations consisted in the main of packages which were already available. These packages are described briefly.

Two pieces of classification software were written; a simple perceptron simulator and a K nearest neighbours classifier. These are not described, as they were straightforward pieces of code implementing standard algorithms.

A.1 Calculation of Texture Measures

A.1.1 Serial Code

The first code which was written for the derivation of texture measures was done on a serial machine. We were not aware until later of the unix command `pgmtexture` which calculates cooccurrence based texture measures from a portable grey map (pgm) file. A description of this command may be found in the unix manual.

The images which are input to the program were plain ASCII files, each character representing a greyscale value $\in [0, 255]$. Each such image file represents an area of more then 1000×1000 pixels. Given these input files, the program is designed to

- Read in the image
- Decompose the image into sub blocks of a specified size, rejecting any additional "edge" regions. To do this it was necessary to convert the 1D input into 2D .
- Rescale the pixel values in each sub block to the specified mean, and to lie in $[0, 15]$.
- Determine the cooccurrence matrix for each sub block, and the texture measures from this matrix.

In the serial program, we used fixed values of displacement in determining the cooccurrence matrices and measures; the angles were $0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}$.

The software is successful in meeting these objectives. However, we decided to convert it to run on a parallel (SIMD) machine, the Thinking Machines Corporation CM-200. This was done because of the greater speed of that machine and to enable us to use the DataVault fast access storage device. This allows parallel variables (the images) to be written and retrieved as single objects, rather than as 1D streams of data.

A.1.2 C* code on the CM-200

Converting the serial code described above for the Connection Machine was straightforward. The language C*, a parallel version of C, was used as being most compatible with the original code, already written in C.

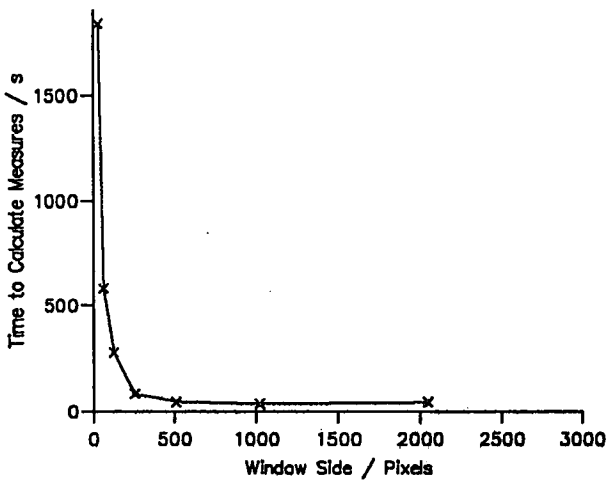

Figure A.1. Time taken to derive the image measures on the CM-200. The image used had 2048 × 2048 pixels so the largest window is the whole image. For windows smaller than 64 × 64 pixels, the program becomes very inefficient as not all the processing elements are being used.

The opportunity was taken to provide some additional features:

- Input was allowed either from an ordinary ASCII file or from the Datavault.
- Cooccurrence matrices could be saved to the DataVault. Further software can be written to display these matrices (as in [20, p. 59-61].)
- The window size was required to be a power of 2, because in the C* language all the dimensions of variables must be powers of 2. Periodic boundary conditions for calculating the cooccurrence matrices were easy to imlement using C*.

- Coarse graining of the images was allowed.

Timings are shown in Figure A.1 for the derivation of texture measures using various window sizes on the CM-200. The image, consisting of 2048×2048 pixels, was stored on the DataVault. It was necessary to write a simple loader program which reads in an image from an ASCII file and write it to the Datavault. This is slower than reading an image directly from the DataVault, but it only has to be done once.

A.2 Other Software

Three packages were all used for neural network training.

- **Rhwydwaith.** This is a neural network simulator written for Meiko computing surfaces. During the course of the research described in this thesis, Rhwydwaith ceased to be supported by the Edinburgh Parallel Computing Centre (EPCC). The package is described in in [118].

- **Neurosys.** This package, written in C for parallel systems supporting the CHIMP message passing system, is the successor to Rhwdwaith.

- **NevProp.** This package was written by Phil Goodman, David Rosen and Allen Plummer. It implements the Fahlman's QuickProp algorithm. This is serial code, which we ran on various Sun systems.

All of these packages are efficient and user friendly. Although no longer supported, Rhwydwaith provides a fast and flexible network imlementation on parallel machines. This is also true of NevProp. For large problems, parallel machines are ideal. Such problems may be decomposed in two ways, either by allocating a number of training patterns to each processor and combining the updates (for

batch learning) or by allocating different network nodes to different processors. Which of these approaches is chosen depends on the type of network, the size of the problem and the characteristics of the machine. However, using a parallel rather than a serial machine introduces the need for communications between processors. These communications require time, and so slow down training. However, the amount of communication needed does not increase much with the size of the problem; but a serial machine will take longer as the problem becomes larger. Thus, parallel machines are more suited to very large problems. The training which we performed did not require the use of a parallel machine.

Appendix B

Images Used in Training

This Appendix contains the photographic images which we used in our training. Each is classified as either a sandstone or a carbonate (limestone.) The images are reproduced as 1.25 times full size; they are presented in monochrome rather than colour. One of the stones, Dunhouse Buff, was available from photographs taken locally and from [94]. The two versions are both given. Unfortunately, the quality of some of the scanned images is not very good, partly because of the scanning process (to have scanned the at a higher resolution would have required a great deal of memory to hold the images) and partly because of the printing process.

Photographs B.1, B.3, B.2, B.14, B.15, B.16 and B.17 were made available by the Department of Petroleum Engineering at Heriot-Watt University, to whom I wish to express my thanks. Figure B.4 was generated by the method of Smith and Freeze[127] using software written by Gillian Pickup of the same department.

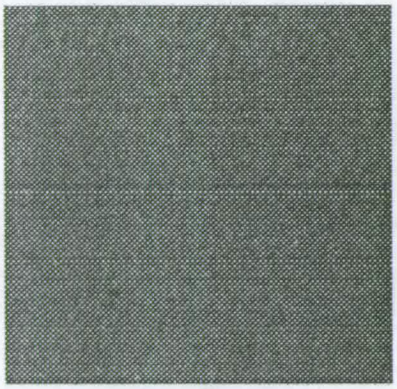

Figure B.1. Gatelawbridge Stone (Sandstone).

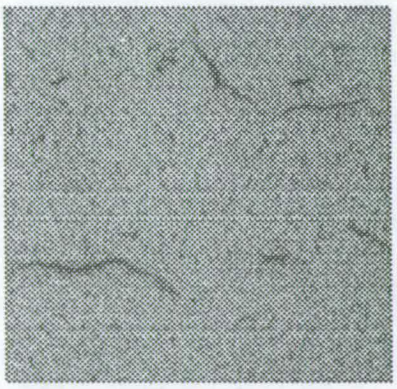

Figure B.2. Dunhouse Buff Stone (Sandstone).

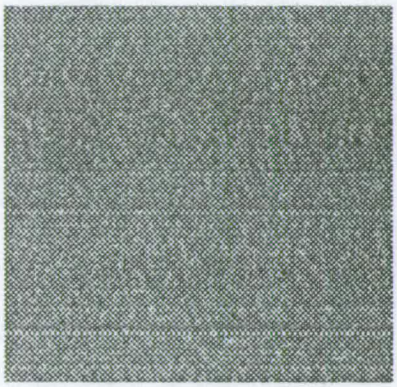

Figure B.3. Locharbriggs Stone (Sandstone).

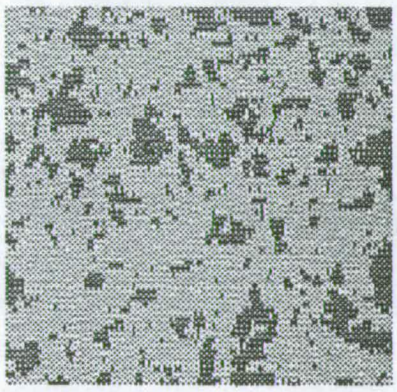

Figure B.4. Stochastic Texture (Artificial).

Figure B.5. Grenoside Stone (Sandstone).

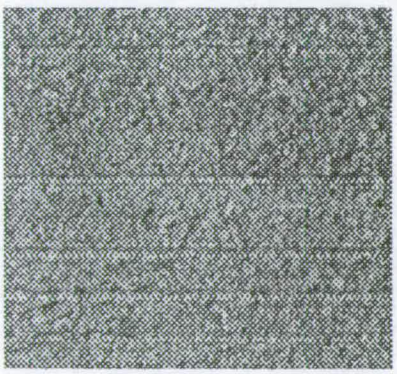

Figure B.6. Bramley Fall Stone (Sandstone).

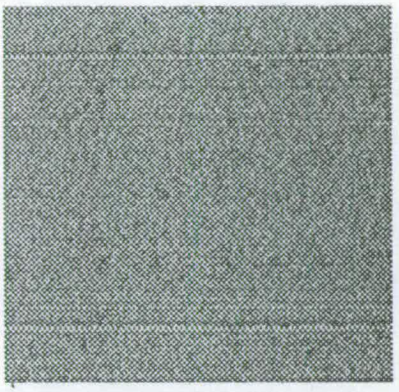

Figure B.7. Elland Edge Flagrock (Sandstone).

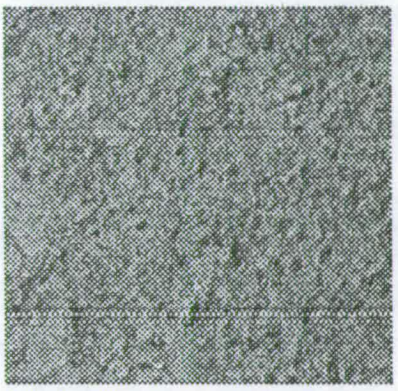

Figure B.8. Fly Delph (Sandstone).

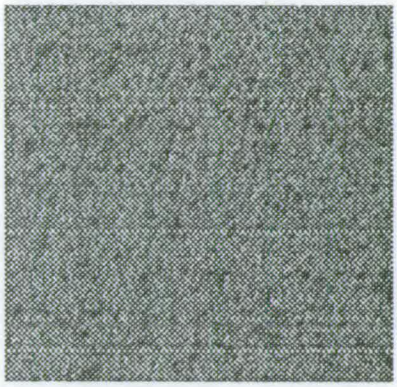

Figure B.9. Hall Dale Stone (Sandstone).

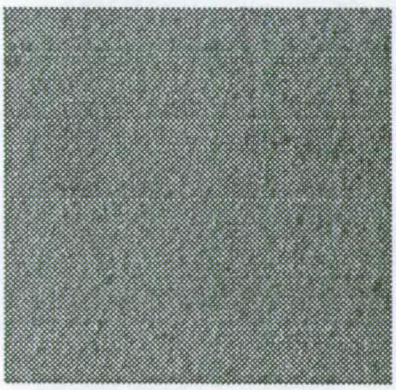

Figure B.10. Dunhouse Buff Stone (Sandstone).

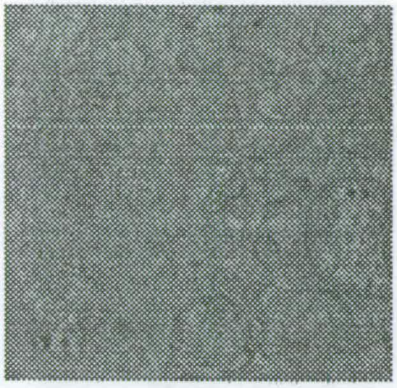

Figure B.11. Purbeck Marble (Carbonate).

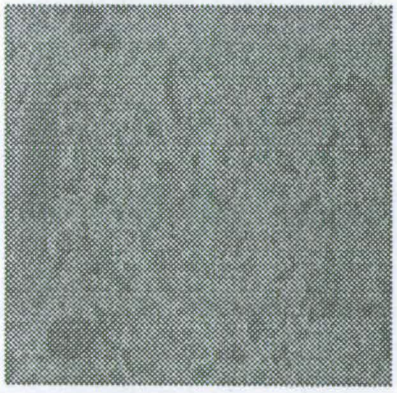

Figure B.12. Hoptonwood Stone (Carbonate).

Figure B.13. Ham Hill Stone (Carbonate).

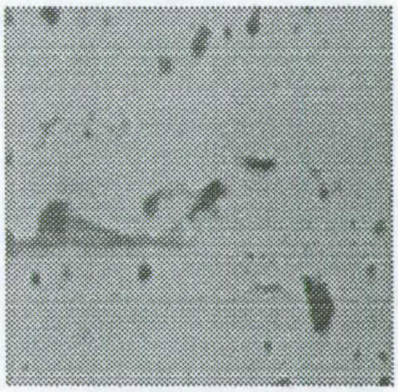

Figure B.14. Tufa (Carbonate).

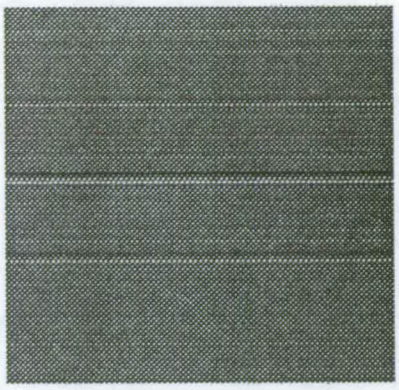

Figure B.15. Beatrice Core (Sandstone).

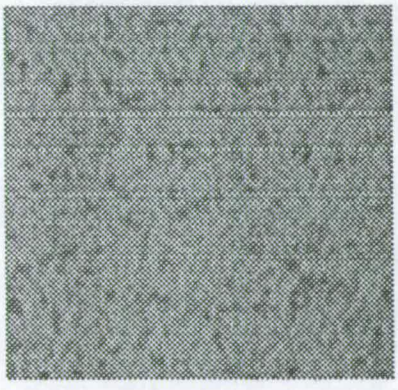

Figure B.16. Mottled Sandstone.

Figure B.17. Liesegang Ring Stone (Sandstone).

Bibliography

[1] G Arfken. *Mathematical Methods for Physicists.* Academic Press, Orlando, Florida, 1985.

[2] J L Baldwin and R M Bateman. Application of a neural network to the problem of mineral identification from well logs. In *Proceedings of the Third International Symposium on Borehole Geophysics for Minerals, Geotechnical and Groundwater Applications*, Las Vegas, Nevada, October 1989.

[3] E B Baum and D Haussler. What size net gives valid generalisation? In *Advances in Neural Information Processing Systems*, pages 81–90, Denver, 1989. Morgan-Kaufman.

[4] R J Baxter. *Exactly Solved Models in Statistical Mechanics.* Academic Press, London, 1982.

[5] R Beale and T Jackson. *Neural Computing: An Introduction.* Adam Hilger, Bristol, 1990.

[6] J Beck and B Ambler. Discriminability of differences in line slope and in line arrangement as a function of mask delay. *Percept. Psychophys.*, 12:33–38, 1972.

[7] K Binder, editor. *Monte Carlo Methods in Statistical Physics.* Springer-Verlag, Berlin, 1979.

[8] C Bing and D M Titterington. Neural networks and statistical perspectives. University of Glasgow preprint. Draft chapters kindly provided by the authors., 1993.

[9] H Bischof, W Schneider, and A J Pinz. Multispectral classification of landsat-images using neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 30(3):482–490, May 1992.

[10] C Bishop. Neural network techniques for pattern recognition. Draft chapters from forthcoming book kindly provided by the author.

[11] H D Block. The perceptron: a model for brain functioning. *Rev. Mod. Phys.*, 34(1):123–135, 1962.

[12] H Bohr, J Bohr, S Brunak, R M J Cotterill, B Lautrup, L Nørskov, O H Olsen, and S B Petersen. Protein secondary structure and homology by neural networks: The α-helices in rhodopsin. *FEBS Letters*, 241:223–228, 1988.

[13] D M Booth. Survey of statistical texture measures. Technical Report RIPRREP/1000/37/88, Research Initiative in Pattern Recognition, Royal Signals and Radar Research Establishment, Malvern, Worcestershire, UK, 1988.

[14] R N Bracewell. *The Fourier Transform and its Applications.* McGraw-Hill, Singapore, 1965.

[15] P Brodatz. *Textures: A Photographic Album for Artists and Designers.* Reinhold, New York, 1968.

[16] S Brunak and B Lautrup. Liniedeling med et neuralt neværk. *Skrifter for Anvendt Matematik og Lingvistik*, 14:55–74, 1989.

[17] S Brunak and B Lautrup. *Neural Networks: Computers with Intuition.* World Scientific, Singapore, 1990.

[18] A E Bryson and Y C Ho. *Applied Optimal Control.* Blaisdell, New York, 1969.

[19] J M Carstensen. Cooccurrence feature performance in texture classification. Preprint kindly made available by the author over Internet.

[20] J M Carstensen. *Description and Simulation of Visual Texture.* PhD thesis, Technical University of Denmark, 1992.

[21] C K Chui. *Introduction to Wavelets.* Academic Press, San Diego, 1992.

[22] R W Conners. Towards a set of statistical features which measure visually perceivable qualities of scenes. In *Proceedings of IEEE Pattern Recognition and Image Processing Conference*, pages 382–390, Chicago, August 1979.

[23] R W Conners. Discriminating textures which have identical second order probabilities using the spatial grey level dependence method. Technical Report 402.82, Louisiana State University Department of Electrical and Computer Engineering, 1982.

[24] R W Conners et al. Segmentation of a high resolution urban scene using texture operators. *Computer Vision Graphics and Image Processing*, 25:273–310, 1984.

[25] R W Conners and C A Harlow. Towards a structural texture analyser based on statistical methods. *Computer Graphics and Image Processing*, 12:224–256, 1980.

[26] Thinking Machines Corporation. *Getting Started in C*.* Thinking Machines Corporation, Cambridge, Massachusetts, 1991.

[27] C Cortes, L D Jackel, S A Solla, V Vapnik, and J S Denker. Learning curves: Asymptotic values and rate of convergence. Presented at the Snowbird neural networks conference. No proceedings published - paper kindly supplied by the authors., 1993.

[28] G W Cottrell, P Munro, and D Zipser. Learning internal representations from gray-scale images: An example of extensional programming. In *Ninth Annual Conference of the Cognitive Science Society*, pages 462–473, Seattle 1987, 1987. Lawrence Erlbaum, Hillsdale.

[29] T M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, 14:326–334, 1965.

[30] T Cressor and P Luner. Description of the spatial grey level dependence method algorithm. *TAPPI Journal*, 73(12):220–222, 1990.

[31] G Cybenko. Continuous valued neural networks with two hidden layers are sufficient. Technical report, Department of Computer Science, Tufts University, Medford, MA, 1988.

[32] G Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[33] E M Darling. Pattern recognition from satellite altitudes. *IEEE Trans. Syst. Sci. Cybern.*, SSC-4:38–47, 1968.

[34] N Dodd. Texture discrimination using multi layer perceptrons. Technical Report RIPRREP/1000/15/87, Research Initiative in Pattern Recognition, Royal Signals and Radar Research Establishment, Malvern, Worcestershire, UK, 1987.

[35] W J M Epping and A R L'Istelle. A neural data compression procedure for seismic images. In *Proceedings of the European Conference on Artificial Intelligence in Petroleum Exploration and Production*, Rueil Malmaison, France, October 1991.

[36] B K Ersbøll and K Conradsen. Automated grading of wood-slabs: The development of a prototype system. Technical Report 10/1991, The Institute of Mathematical Statistics and Operations Research, The Technical University of Denmark, DK-2800, Lyngby, Denmark, 1991.

[37] S E Fahlman. An empirical study of learning speed in back propagation networks. Technical Report CMU-CS-88-162, Carnegie-Mellon University, 1988.

[38] S E Fahlman. Faster-learning variations on back propagation: An empirical study. In *Proceedings of the 1988 Connectionist Models Summer School.* Morgan Kaufmann, San Mateo, 1988.

[39] S E Fahlman and C Lebiere. The cascade-correlation learning architecture. In D S Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 524–532, Denver, 1990. Morgan Kaufmann, San Mateo.

[40] C L Farmer. Numerical rocks: The mathematical generation of resevoir geology. Presented at the joint IMA/SPE European Conference on the Mathematics of Oil Recovery, Robinson College, Cambridge University, July 1989.

[41] O D Faugeras and W K Pratt. Decorrelation methods of texture feature extraction. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-2(4):323–332, July 1980.

[42] J Feder. *Fractals.* Plenum Press, New York, 1988.

[43] R P Feynman. *Statistical mechanics - A Set of lectures.* Addison-Wesley, Redwood City, 1972.

[44] D L Filkin. Distributed parallel processing network wherein the connection weights are generated using stiff differential equations, September 1991. US Patent Number 5046020.

[45] R A Fisher. The use of multiple measurements in taxonomic problems. *Ann. Eugenics*, 7:179–184, 1936.

[46] I Fogel and D Sagi. Gabor filters as texture discriminator. *Biological Cybernetics*, 61:103–113, 1989.

[47] M R Frean. *Small Nets and Short Paths: Optimising Neural Computation.* PhD thesis, The University of Edinburgh, Edinburgh, UK, 1990.

[48] M R Frean. A "thermal" perceptron learning rule. *Neural Computation*, 4(6):946–957, 1992.

[49] D Gabor. Theory of communication. *Proc. IEE*, 93(26):429–441, 1946.

[50] A Gagalowicz. A new method for texture fields synthesis: Some applications to the field of human vision. *IEEE Trans. Pattern Analysis and Machine Intellingence*, PAMI-3:520–533, 1981.

[51] S I Gallant. Optimal linear discriminants. In *Eighth International Conference on Pattern Recognition*, pages 849–852, Paris 1986, 1986. IEEE, New York.

[52] S I Gallant. Three contructive algorithms for network learning. In *Proceedings of 8th Annual Conference of the Cognitive Science Society*, pages 652–660, 1986.

[53] S I Gallant. Perceptron based learning algorithms. Technical Report NU-CCS-89-21, College of Computer Scienc, Northeastern University, Boston, MA, 1989.

[54] E Gardner. Maximum storage capacity in neural networks. *Europhysics Letters*, 4:481–485, 1987.

[55] E Gardner, N Stroud, and D J W Wallace. Training with noise and the storage of correlated patterns in a neural network model. *J. Phys. A*, 22:2019–2030, 1989.

[56] G Geiger and J E Kogler. Scaling images and image features via the renormalisation group. Preprint kindly made available by the authors over Internet., 1993.

[57] J J Gibson. *The Perception of the Visual World.* Houghton-Mifflin, Boston, USA, 1950.

[58] R P Gorman and T J Sejnowski. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*, 1:75–89, 1988.

[59] R P Gorman and T J Sejnowski. Learned classification of sonar targets using a massively-parallel network. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36:1135–1140, 1988.

[60] H Y Greenspan and R Goodman. Remote sensing image analysis via a texture classification neural network. 1992.

[61] Z Q Gu, C N Duncan, P M Grant, C F N Cowan, E Renshaw, and M A Mugglestone. Textural and spectral features as an aid to cloud classification. *International Journal of Remote Sensing*, 12(5):953–968, 1991.

[62] P Guttorp and A Nalden, editors. *Statistics in the Environmental and Earth Sciences.* E. Arnold, 1992.

[63] H H Haldorsen and E Damsleth. Stochastic modelling. *Journal of Petroleum Technology*, pages 404–412, April 1990.

[64] Y J Han and J C Hayes. Soil cover determination by image analysis of textural infomation. *Transactions of the ASAE*, 33:681–686, 1990.

[65] P Hanks, editor. *Collins Dictionary of the English Language.* Collins, Glasgow, 1979.

[66] R M Haralick. Statistical and structural approaches to texture. *Proc. IEEE*, 69:786–804, 1979.

[67] R M Haralick et al. Textual features for image classification. *IEEE Trans. on Systems, Man and Cybernetics*, SMC-3:610–621, November 1973.

[68] R M Haralick and K Shanmugan. Computer classification of resevoir sandstones. *IEEE Trans. Geosci. Electron.*, GE-11:171–177, October 1973.

[69] H H Hardy. The fractal character of photos of slabbed cores. *Mathematical Geology*, 24(1):73–97, 1992.

[70] D A Harris, J J M Lewis, and D J Wallace. The identification of lithofacies types in geological imagery using neural networks. In *Proceedings of the European Conference on Artificial Intelligence in Petroleum Exploration and Production*, Aberdeen, 1993.

[71] J K Hawkins. *Texture Properties for Pattern Recognition.* Academic Press, New York, 1970. In: Picture Processing and Psychopictorics, ed Lipkin, B C and Rosenfeld,A.

[72] W D Hillis. *The Connection Machine.* MIT Press, Cambridge, Massacheusetts, 1985.

[73] R Hofmann. Neurosys user guide: A portable library for feedforward neural networks. Technical report, Edinburgh Parallel Computing Centre, University of Edinburgh, Edinburgh, UK, 1992.

[74] K Hornik, M Stinchcombe, and H White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[75] J N Hwang and E T Chen. Textured image synthesis and segmentation via neural network probabilistic modelling. Technical report, University of Washington, Information Processing Laboratory, Department of Electrical Engineering, Seattle, USA, 1993. Kindly made available over Internet.

[76] R A Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1:295–307, 1988.

[77] F James. Monte carlo theory and practice. *Rep. Prog. Phys.*, 43:1145–1189, 1980.

[78] B Julesz. Visual pattern discrimination. *IRE Trans. Inf. Theory*, IT8:84–92, 1962.

[79] B Julesz. *Cluster formation at Various Perceptual Levels.* Academic Press, New York, 1969. In: Methodologies of Pattern Recognition, ed Watanabe, S.

[80] B Julesz. Experiments in the visual perception of texture. *Scientific American*, 232:34–43, 1975.

[81] B Julesz. Spatial nonlinearities in the instantaneous perception of textures with identical power spectra. *Philos. Trans. Roy. Soc. London B*, 290:83–94, 1980.

[82] B Julesz. Textons, the elements of texture perception and their properties. *Nature*, 290:91–100, 1981.

[83] B Julesz and T Caelli. On perceptual analysers underlying visual texture discrimination 1. *Biological Cybernetics*, 28:167–175, 1978.

[84] B Julesz and T Caelli. On perceptual analysers underlying visual texture discrimination 2. *Biological Cybernetics*, 29:201–214, 1978.

[85] B Julesz et al. Inability of humans to distinguish between visual textures that agree in second order statistics - revisited. *Perception*, 2:391–405, 1973.

[86] B Julesz et al. Visual discrimination of textures with identical third order statistics. *Biological Cybernetics*, 31:137–140, 1978.

[87] P R Krishnaiah and L N Kanal. *Handbook of Statistics: Classification, Pattern Recognition and Reduction of Dimensionality*, volume 2. North-Holland, 1982.

[88] A Krogh and J A Hertz. A simple weight decay can improve generalisation. Preprint kindly made available by the authors over Internet.

[89] J Krogh, A Hertz, and R G Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwood City, CA, 1991.

[90] A Lapedes and R Farber. Nonlinear signal processing using neural networks: Prediction and system modelling. Technical Report LA–UR–87–2662, Los Alamos National Laboratory, Los Alamos, NM, 1987.

[91] Y Le Cun, B Boser, J S Denker, D Henderson, R E Howard, W Hubbard, and L D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[92] Y Le Cun, B Boser, J S Denker, D Henderson, R E Howard, W Hubbard, and L D Jackel. Handwritten digit recognition with a back-propagation network. In D S Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 396–404, Denver 1989, 1990. Morgan Kaufmann, San Mateo.

[93] E Leary. *The Building Limestones of the British Isles.* HMSO, London, 1983.

[94] E Leary. *The Building Sandstones of the British Isles.* HMSO, London, 1984.

[95] M R Leeder. *Sedimentology: Process and Product.* HarperCollins Academic, London, 1982.

[96] S Lipshultz. *Finite Mathematics.* MacGraw-Hill, New York, 1966.

[97] J D McCauley, B R Thane, and A D Whittaker. Fat estimation in beef ultrasound images using texture and adaptive logic networks. *Transactions of ASAE.* Kindly made available by the authors over Internet.

[98] J L McClelland, D E Rumelhart, et al. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, Cambridge, MA, 1986.

[99] M Mézard and J P Nadal. Learning in feedforward layered networks: The tiling algorithm. *Journal of Physics A*, 22:2191–2204, 1989.

[100] M Minsky and S Pappert. *Perceptrons.* MIT Press, Cambridge, Massachusettts, expanded edition, 1988.

[101] O R Mitchell. Image segmentation using a local extrema texture measure. *Pattern Recognition*, 10:205–210, 1978.

[102] G J Mitchison and R M Durbin. Bounds on the learning capacity of some multi-layer networks. *Biological Cybernetics*, 60:345–356, 1989.

[103] N Morgan and H Bourland. Generalisation and parameter estimation in feedforward nets: Some experiments. Technical Report TR-89-017, International Computer Science Institute, Berkeley, CA, 1989.

[104] E Mousset. Spikes filtering with neural networks: A two-stage detection system. *Revue de l'institute Francais du Pètrole*, 47(3):407–421, 1992.

[105] R M Neal. *Probabilistic Inference Using Markov Chain Monte Carlo Methods.* PhD thesis, University of Toronto, 1993. Department of Computer Science Technical Report CRG-TR-93-1.

[106] S J Nowlan and G E Hinton. Simplifying neural networks by soft weight sharing. *Neural Networks*, 4;473–493, 1992.

[107] E Oja, J Parkinnen, et al. *From Pixels to Features*, chapter Regularity Measurement, Classification, and Segmentation of Textures. Elsevier Science Publishers B.V. (North-Holland), 1989. Ed. Simon,J C.

[108] D A Osborne. Neural networks provide more accurate resevoir permeability. *Oil and Gas Journal*, 1992. September 28th.

[109] A J Owens and D L Filkin. Efficient training of the back propagation network by solving a system of stiff ordinary differential equations. In *Proceedings of International Joint Conference on Neural Networks*, volume 2, pages 381–386, June 1989. Washington DC.

[110] D B Parker. Learning logic. Technical Report TR–47, Center for Computational Research in Economics and Management Science, MassachusettsInstitute of Technology, Cambridge, MA, 1985.

[111] S H Peckinpaugh. An improved method for computing grey level cooccurrence based texture measures. *Computer Vision Graphics and Image Processing*, 53:574–580, 1991.

[112] A P Pentland. Fractal based description. In *Proceedings of the International Joint Conference on Pattern Recognition*, pages 973–981, Karlsruhe, August 1983.

[113] R M Pickett. *Visual Analysis of Texture in the Detection and Recognition of Objects.* Academic Press, New York, 1970. In: Picture Processing and Psychopictorics, ed Lipkin, B C and Rosenfeld, A.

[114] D A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D S Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1, pages 305–313, Denver 1988, 1989. Morgan Kaufmann, San Mateo.

[115] W K Pratt. *Digital Image Processing.* Wiley - Interscience, New York, 1978.

[116] W K Pratt et al. Visual discrimination of stochastic texture fields. *IEEE Trans. Systems, Man and Cybernetics*, SMC-8:796–804, 1978.

[117] N Qian and T J Sejnowski. Predicting the secondary structure of globular proteins using neural network models. *Journal of Molecular Biology*, 202:865–884, 1988.

[118] G D Richards. Documentation for rhwydwaith. Technical Report ECSP-UG-7, Edinburgh Parallel Computing Centre, Edinburgh, UK, 1990.

[119] G D Richards. *Implementation and Capabilities of Layerd Feedforward Networks.* PhD thesis, The University of Edinburgh, Edinburgh, UK, 1990.

[120] B D Ripley. Classification and clustering in spatial and image data. In Martin Schader, editor, *Analysing and Modelling Data and Knowledge - Proceedings of the 15th Annual Conference of the Gesellschaft fur Klassifikation e.V.*, University of Saltzburg, 1992. Spinger-Varlag, Berlin.

[121] T Robinson. Practical network design and implementation. Technical report, Cambridge University Engineering Department, Cambridge, UK, 1992. Presented at Cambridge Neural Network Summer School. Kindly made available by the author over Internet.

[122] M Rosenblatt. *Principles of Neurodynamics.* Spartan Books, New York, 1962.

[123] M Rosenblatt and D J Slepian. *J. Soc. Ind. Appl. Math*, 10:537–549, 1962.

[124] F Schnorrenberg, S K Aityan, A R Khachatrayan, and R A Startzman. Fracture recognition with neural networks in high-resolution borehole televiewer data. In *Proceedings of the Conference on Artificial Intelligence in Petroleum Exploration and Production*, 1993.

[125] T J Sejnowski and C R Rosenberg. Parallel networks that learn to pronounce english text. *Complex Systems*, 1:145–168, 1987.

[126] S A Shearer and R G Holmes. Plant identification using colour cooccurrence matrices. *Transactions of the ASAE*, 33:2037–2044, 1990.

[127] L Smith and R A Freeze. Stochastic analysis of steady state groundwater flow in a bounded domain 2: Two dimensional simulations. *Water Resources Research*, 15, December 1979.

[128] M Smith, N Carmichael, I Reid, and C Bruce. Prediction of lithofacies types and qualities using a distributed neural network. Technical Report EPCC-TR91-10, Edinburgh Parallel Computing Centre, Edinburgh, UK, 1991. Published in Proceedings of IEEE Workshop: Neural Networks for Signal Processing.

[129] S A Solla. Capacity control in classifiers for pattern recognition. In S K Kung et al., editors, *Neural Networks for Signal Processing 2*, pages 255–266, 1992. Proceedings of IEEE Workshop.

[130] H E Stanley. *Introduction to Phase Transitions and Critical Phenomena.* OUP, Oxford, 1971.

[131] R H Swendsen. Monte carlo calculation of renormalised coupling parameters. *Phys. Rev. Lett.*, 52(14):1165–1168, April 1984.

[132] R H Swendsen. Monte carlo calculation of renormalised coupling parameters - 1. *Phys. Rev. B*, 30(7):3866–3874, 1984.

[133] R H Swendsen. Monte carlo calculation of renormalised coupling parameters - 2. *Phys. Rev. B*, 30(7):3875–3881, 1984.

[134] R H Swendsen. Monte carlo renormalisation group. *J. Stat. Phys*, 34:963–976, 1984.

[135] G Tesauro. Neurogammon wins computer olympiad. *Neural Computation*, 1:321-323, 1990.

[136] T Tollenaere. Supersab: Fast adaptive back propagation with good scaling properties. *Neural Networks*, 3:561–573, 1990.

[137] T Tollenaere. *The Simulation of Entropy Driven Artificial Neural networks on Message Passing Multiprocessors.* PhD thesis, Catholic University of Leuven, Leuven, Belgium, March 1993.

[138] M R Turner. Texture discrimination by gabor functions. *Biological Cybernetics*, 55:51–82, 1986.

[139] A S Weigend, D E Rumelhart, and B A Huberman. Predicting the future: A connectionist approach. In T J Sejnowski, G E Hinton, and D S Touretzky, editors, *Proceedings of the 1990 Connectionist Models Summer School*, San Mateo, CA, 1990. Morgan-Kaufmann.

[140] P Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, 1974.

[141] A D Whittaker, Bo S Park, and J D McCauley. Ultrasonic signal classification for beef quality grading through neural networks. In *Proceedings of the Automated Agriculture for the 21st Century Symposium*, December 1991. Kindly made available by the authors over Internet.

[142] J W Woods. Two dimensional discrete markov random fields. *IEEE Trans. Inform. Theory*, IT-18(2):232–240, March 1972.

[143] F Y Wu. The potts model. *Rev. Mod. Phys*, 54(1):235–268, 1982.

[144] L Wu and F Fallside. Fully vector quantised neural network based code excited nonlinear prediction speech coding. Technical report, Cambridge

University Engineering Department, Cambridge, UK, March 1992. Submitted to IEEE Trans. on Signal Processing.

[145] C Zhou and X Wu. Neural network based formation analysis estimation from well logs in quantitative log analysis: A comparitive study. In *Proceedings of the SPE Asia Pacific oil and Gas Conference*, Singapore, February 1993.