Image Database Retrieval Using Neural Networks

Richard Rickman

Brunel
THE UNIVERSITY OF WEST LONDON

Image Database Retrieval Using Neural Networks

Richard Rickman

Brunel

THE UNIVERSITY OF WEST LONDON

Image Database Retrieval Using Neural Networks

A Thesis submitted for the degree of Doctor of Philosophy

by

Richard Matthew Rickman

Neural Networks and Pattern Recognition Group
Dept. Electrical & Electronic Engineering
Brunel University
Uxbridge
Middx.

July 1993

Abstract.

Richard Rickman
Neural Networks and Pattern Recognition Group
Dept. Electrical & Electronic Engineering
Brunel University
Uxbridge
Middx.

The broad objective of this work has been to achieve retrieval of images from large unconstrained databases using image content. The problem is typified by the need to locate a target image within a database where no numerical indexing terms exist. Here, retrieval is based on important features within in an image and uses sample images or user sketches to specify a query. A typical query might be framed as "Find all images similar to this one", for example. The aim of this work has been to show how neural networks can provide a practical, flexible and robust solution to this problem.

A neural network is basically an adaptive information filter which can be used to extract the salient characteristics of a data set during a training phase. The transformation learnt by the network can map the images into compact indices which support very rapid fuzzy matching of images across the database. This learning process optimises the performance of the code with respect to the contents of the database.

We assess the applicability of several neural network architectures and learning rules for a practical coding scheme and investigate how the system parameters affect the performance of the system. We introduce a novel learning law which has a number of advantages over existing paradigms. In-depth mathematical analysis and extensive empirical tests are used to corroborate the arguments presented throughout.

This thesis aims to show the nature of the image retrieval problem, how current research trends attempt to tackle it and how neural networks can offer us a real alternative to conventional approaches.

Acknowledgements

I would like to thank my friends within the Neural Networks and Pattern Recognition group at Brunel who made this project so stimulating and rewarding. My special thanks go out to my supervisor, Professor T. John Stonham, whose constant encouragement helped me to realise that the darkest hour comes just before the dawn.

This work was funded by the British Library Research and Development Department research contract MO921.

Contents

Abstract. i

Acknowledgements ii

Chapter 1 Introduction 1

1.1. Motivation for Research 1

1.2 Objectives 1

1.3. Applying Neural Networks to Practical Image Database Systems. 2

1.4. Overview of the Thesis. 3

Chapter 2 The State-of-the-art in Image Database Retrieval Systems 7

2.1. Introduction. 7

2.2. The Growth of Image Database Technology. 8

2.3. Contrasting Conventional and Image Databases. 9

2.4. Elements of a Content-Based Retrieval Mechanism. 11

2.5. Shape and Feature Representation in an Image Index 13

2.5.1. Global and local features for indexing 13

2.5.2. Spatially dislocated features as indices 14

2.5.3. Object/position descriptors as indices. 14

2.5.4. Low level features as indices. 14

2.6. Easing the Burden of Object Recognition for Index Construction. 15

2.6.1. Interactive object recognition 15

2.6.2. Indexing CAD drawings 15

2.7. Similarity Measures for Image Retrieval Systems 16

2.8. Query Processing. 17

2.8.1. Query by sketch and images 18

2.8.2. Query by colour 18

2.8.3. Query by subjective descriptions. 18

2.9. Data Representation and Organisation 19

2.10. The role of Neural Networks in IDBMS. 21

2.11. The Nature of Queries in a Typical Commercial Pictorial Archive 23

2.10.1. Characterisation of requests 23

2.12. Discussion 25

Chapter 3 The Fundamentals of Image Coding using Neural Networks. 28

3.1. Introduction 28

3.2. Neural Network Research - Some Key Issues. 29

3.2.1. The 'curse of dimensionality' and the blessing of the neuron. 29

3.2.2. The evolution of the species. 29

3.2.3. A simple coding scheme using neural networks. 32

3.2.4. What perceptrons cannot do 33

3.2.5. Backpropogation for image coding 34

3.3. PCA and its Relationship to Our Coding Scheme. 36

3.4. Unsupervised Learning in Neural Networks 37

3.4.1. What features should our coding network learn ? 37

3.4.2. Hebb's law - a biological paradigm for unsupervised learning. 38

3.4.3. Learning to forget 40

3.4.4. Constrained Hebbian learning - a generalised approach. 40

3.4.5. The relationship between constrained Hebbian learning and Principal Component Analysis. 42

3.4.6. Implementing a multiple Principal Component system on a neural network. 42

3.5. Non-Linearities and Multiple-layer Networks - Are they Appropriate? 43

3.5.1. Is one layer enough ? 44

3.6. Discussion 44

Chapter 4 An Empirical Study of the System Parameters for a Sum-of-weights type Neural Network 46

4.1. Introduction 46

4.2. An Image Coding Scheme using Constrained Hebbian Learning 47

4.2.1. Does this strategy produce a conformal mapping? 47

4.2.2. Learning rate and the dynamic behaviour of the constrained Hebbian node 48

4.2.3. A practical test to assess the performance of an image retrieval mechanism. 48

4.2.4. How does order of presentation affect training? 49

4.2.5. Dimensionality and learning dynamics. 51

4.2.6. Data set size and learning dynamics. 52

4.3. A Multi-Node Coding Scheme using Constrained Hebbian Learning. 53

4.4. Database Contents and System Performance. 55

4.4.1. Matching code length to diversity of database 55

4.4.2. The need for re-training. 58

4.4.3. Re-training for a database containing limited image types......58
4.4.4. Re-training for a database containing unlimited image types......59
4.5. Non-Linearities - A Practical Assessment of Their Worth......60
4.5.1. Non-linearities and the prospect for improved performance......60
4.6. An Intelligent Video Editing System using the Neural Network Coding Scheme......64
4.6.1. Methodology......64
4.6.2. Results......65
4.6.3. The effect of the number of nodes on the performance of the editing system......68
4.7. Discussion......69
Chapter 5 Logical Neural Networks......73
5.1. Introduction......73
5.2. Why use Logical Neurons ?......73
5.3. The WISARD architecture......74
5.4. How does the tuple size affect the performance of a Logical Neural Network ?......75
5.4.1. The nodal transformation of an n-tuple......75
5.4.2. Response profiles and tuple size - an analysis......76
5.4.3. The contrasting roles of image classification and coding systems......78
5.4.4. The effect of tuple size for a simple coding scheme......80
5.5. Training and System Performance......81
5.5.1. Training and tuple size - an analysis......81
5.5.2. Training by image shifting for a practical coding scheme......84
5.5.3. Training by adding noise to an image for a practical coding scheme......86
5.6. Mapping the Tuples onto the Image......87
5.6.1. Why do some tuple mappings perform better than others?......87
5.6.2. The tuple as a feature extractor - what's going on ?......90
5.6.3. Mapping tuples to maximise the discriminant information......92
5.7. A Multi-Node encoding Scheme using Logical Neural Networks......93
5.7.1. Matching the size of the code to the diversity of the database......94
5.7.2. The effect of training in a multi-node coding scheme......96
5.8. An unsupervised Coding Scheme using a Logical Neural Network......98

5.9. Discussion 102

Chapter 6 A Novel Self Organising Learning Law for Image Coding. 105

6.1. Introduction 105

6.2. An Overview of Self -Organisation in Logical Neural Networks........ 106

6.2.1. The Feature Sensitive Node. 106

6.2.2. Tamboratzis' SOLNN 107

6.2.3. Allinsons' logical Kohonen clone........ 108

6.2.4. A further logical implementation of a Kohonen feature map. 110

6.2.5. On the formalism of learning paradigms for LNNs. 110

6.3. Hebbian learning for Logical Neural Networks. 112

6.3.1. Discriminant features in Self Organising Logical Neural Networks. 112

6.4. A New Self Organising Hebbian Learning Law from First Principles 116

6.4.1. Developing the learning law. 116

6.4.2. Analysing the learning law for a 1-tuple system........ 119

6.4.3. Extending the novel learning law to larger tuple sizes........ 121

6.4.4. Extending the learning law to multi-node systems. 123

6.4.5. Effect of database diversity on performance of a winner-take-all implementation of the learning law. 125

6.4.6. What tuple mapping gives the best performance for the learning law ? 126

6.5. Applying the Learning Rule to a Sum-of-Weights Node. 127

6.5.1. Why is a sum-of-weights implementation preferable to a 1-tuple system ? 128

6.5.2. Orthogonal learning in a sum-of-weights implementation........ 129

6.6. Some Practical Results for the New Learning Law. 130

6.6.1. When should we re-train the neural network? 131

6.7. Comparing the Performance of Four Learning Systems Discussed in this Thesis. 132

6.7.1. Comparing our coding scheme with a mask matching system. 135

6.8. Is a 1-tuple Coding Scheme Really Performing Mask Matching ? 136

6.9. Discussion. 136

Chapter 7 Discussion and Conclusions. 140

7.1. Defining the Problem........ 140

7.2. Existing and Previous Work........ 141

7.3. Basic Thrust of the Neural Network Coding Scheme. 142

7.4. What Kind of Neural Transformation Do We Want ? 143
7.5. Developing a Suitable Learning Law for the Coding Scheme. 143
7.6. Logical Neural Networks for Coding Images. 145
7.7. An Objective Appraisal of LNNs for our Coding Scheme. 147
7.8. When do we Need to Retrain the System?.. 149
7.9. A Coding Scheme Based on Fixed Local Transforms 150
7.10. Epilogue. 152

Bibliography.. 154
Appendix A - Database for the 'Intelligent' Video editor. 162
Appendix B - A Collection of the Author's Papers.. 166

Chapter 1

Introduction.

1.1. Motivation for Research.

We attempt to address the following problem: *We want an image, or something similar to it, from a large database. We think that its in there and we have some idea of the important features which characterise it. How can we access that image?*

1.2 Objectives.

Continuing advances in data storage and transmission technologies mean that Image Database Systems are now a viable prospect. However, the technologies which have been applied so successfully to text database retrieval mechanisms cannot readily be harnessed for image databases. This is because image data is typically richer in information content than simple alpha-numeric text; we have all heard that "a picture paints a thousand words". Image retrieval by content will only become a practical prospect when general purpose automatic feature extraction is possible. This relies heavily upon computer vision and pattern recognition techniques to formulate efficient indexes. Extracting indexing features has proved to be an extremely difficult process to automate using conventional image processing techniques as those features deemed to be important to a human are typically subtle, enigmatic and often not amenable to deterministic analysis.

Feature based indexing for retrieval of information is one of the main differences between image and conventional database systems. Here, retrieval is based on important features within in an image and uses sample images or user sketches, for example, to specify a query. A typical query might be framed as: "Find all images similar to this one".

This thesis aims to show the nature of the image retrieval problem, how current research trends attempt to tackle it and how neural networks can offer us a real alternative to conventional approaches.

1.3. Applying Neural Networks to Practical Image Database Systems.

A neural network is basically an adaptive information filter which can be used to extract the salient characteristics of a data set during a training phase. The transformation learnt by the network can map the images into compact indices which support very rapid fuzzy matching of images across the database. This learning process gives neural networks their remarkable ability to adapt to the nature of the problem in a non-deterministic fashion. The Neural Network Transformation can optimise the efficacy of the indexing scheme according to the distribution of images within the database.

Neural network technology, contrary to popular belief, is not new and has been with us for over half a century. It may have been around for a while but it is still an immature science struggling to establish itself and has yet to find widespread acceptance for practical problems. Neural networks have promised much but, in reality, delivered little. Just why is this?

The non-deterministic aspect of neural network computation is, at once, both a blessing and a curse: it offers the *potential* for finding a mapping from the problem to the solution where no obvious one exists, but it might not find the best one or even an adequate one. Neural network practitioners who seek to find concrete solutions to real-world problems find this rather irksome. Part of the problem lies with the fact that neural networks frequently require a host of often abstruse network parameters to be adjusted for the system to perform the desired function. Such parameters might include the network topology, the learning law 'constants', the nodal functionality and the scope of the training data, for example. This is seldom a straightforward affair and training these systems typically requires expert knowledge both of the problem domain and of neural network systems in general. This has, by and large, restricted their application to rather esoteric problems. This is obviously not acceptable for a practical system.

It is generally held that neural networks will only begin to find favour with a curious public when they can be seen to offer a robust solution to a variety of practical problems. To this end, it is imperative that we develop systems which can be trained with only a modest level of expertise. A primary objective of this work is to investigate how the parameters for the architectures and learning laws discussed here affect the performance of the retrieval mechanism with a view to completely automating this phase of the process. We develop an entirely novel learning law which does not require manual selection of any network parameters.

1.4. Overview of the Thesis.

We investigate the nature of the transformation required by the code and show how it can be supported on a neural network architecture. We give a brief overview of the development of several different network architectures and highlight some of the fundamental issues pertinent to our coding scheme such as network topologies, learning paradigms and node functionality, for example.

We show how the network parameters need to be adjusted as the contents of the database change and investigate methods of automating the selection of these parameters. We assess the applicability of several architectures and learning rules and introduce a novel learning law which has a number of advantages over existing paradigms. In-depth mathematical analysis and extensive empirical tests are used to corroborate the arguments presented throughout.

In chapter 2 we aim to highlight the requirements of a practical Image Database Management System and outline the approaches that have been adopted to solve this difficult problem. We take a closer look at the essential components of a practical Image Database Management System and concentrate on mechanisms which support automatic image retrieval by content. Such a system must make a provision for automatic feature extraction for constructing image indexes, data structures to contain these indexes and query methods to support image access.

Much of chapter 2 is given over to a review current state-of-the-art in automatic image indexing so that the work presented in the rest of this thesis might be put into proper perspective. We summarise the different approaches to feature extraction, the type of features that have been used index images to date and discuss the relative merits of each.

We present an analysis of the type of queries posed to a typical commercial picture archive and show that the subjective nature of these requests are not always amenable to conventional automatic processing.

Chapter 3 demostrates how a simple neural network architecture and associated learning rule can be used as the basis of a very effective coding mechanism for our image retrieval system. We introduce the neural network as an adaptive information filter which can learn the important features within a database of images. We give a brief overview of the fundamental issues in the development of neural networks and highlight some of the factors pertinent to our coding scheme.

We indicate that the coding transform required by our scheme has much in common with a classical statistical analysis technique called Principal Component Analysis. We show how a learning law, found to exist in 'real-life' neurons can be developed and implemented on a single layer feed-forward linear neural network to produce a very efficient approximation of Principal Component Analysis. Much of the mathematical formalisms, insights and theoretical foundations presented in this chapter are applied in a practical context in chapter 4 which provides empirical backing to the work discussed in chapter 3.

In chapter 4 we investigate the factors effecting the performance of a practical retrieval-by-image-content system. Initially, we work with a database of machine printed fonts which permits an objective appraisal of the performance of a neural network based scheme. Extensive practical tests and computer simulations are used to back up the arguments presented throughout.

The rate at which the neuron learns can have a profound effect upon the performance of the system. We investigate how the order of presentation of training data, its size and the size of the data set affect the choice of the learning rate. The performance is also dependent upon the diversity of images within the database - we investigate the extent of this relationship. We show how the code length and nodal transformation may be optimised with respect to the contents of the database. The nodal transformation is changed by re-training the system. We take a close look at the factors affecting the need for re-training.

We provide some theoretical and empirical evidence to show that, though the system is extraordinarily simple, further enhancements are unlikely to improve upon its performance.

We demonstrate how the coding scheme developed here may be used as the basis of an 'intelligent' video editing system.

The objective of chapter 5 is to show how a single layer Logical Neural Network can be used as the basis for a very simple coding scheme and to contrast the performance with the sum-of-weights paradigm discussed in Chapters 3 and 4.

We introduce the WISARD neural network architecture and investigate, both through mathematical analysis and experimental results, how the network parameters effect the performance of the code. Such parameters include tuple size and mapping strategy, training methodology and the number of nodes used. We show how a very basic unsupervised Logical Neural Network coding mechanism can be used to retrieve images from a database of machine printed characters.

In chapter 6 we trace the development of a novel learning paradigm for a Self Organising neural network. The learning law began life as an adjunct to the work on Logical Neural Networks described in chapter 5.

We give a brief overview of existing Self Organising learning paradigms for Logical Neural Networks and assess their suitability for our own particular application. We take an empirical look at the underlying principles governing Self Organisation in these systems and discuss how these might be implemented using a learning law. We go on to develop a mathematical formalism for this law and present an analysis of the system dynamics.

We present a form of this same learning law which can be supported on a sum-of-weights type node and contrast its performance with a Logical Neural Network implementation.

We set out to determine whether Self Organising Logical Neural Network architectures are more appropriate for our particular application than sum-of-weights type counterparts, both in terms of speed of training, memory requirements and optimality of solution. We compare and contrast the performance of four different Self Organising neural network coding schemes for our image retrieval system.

We consolidate the major findings of this research in chapter 7 and highlight potentially fruitful avenues worthy of further investigation.

This thesis encapsulates the findings of a three year British Library funded project to investigate the use of neural networks for image database retrieval systems. It provides a very firm base of practical research upon which to build. This author believes that ambitious demonstrator projects are sorely needed to raise both public awareness and confidence in neural networks. In the final part of this thesis we propose a demonstrator project, based on the Neural Network Transformations investigated in this thesis, which goes some way toward this.

Chapter 2

The State-of-the-art in Image Database Retrieval Systems.

2.1. Introduction.

The technologies which have been developed to transmit, store and access alphanumeric data have now come of age and we are poised on the edge of a new era in information systems design. The next logical step is the design of efficient, flexible and robust systems for handling image data. So, where are they? Why can't the technologies which have been applied so successfully to text-databases be harnessed for image databases, for instance? What is it about image data that makes it so difficult to manage using conventional Database Management System (DBMS) practice?

We attempt to address the following problem: We want an image, or something like it, from a large database. We think that its in there and we have an idea of the important features which characterise that image. How can we get it out?

In this chapter we aim to highlight the requirements of a practical Image Database Management System (IDBMS) and outline the approaches that have been adopted to solve this difficult problem. Such a problem will only really be resolved when our IDBMS can isolate, encode and compare entities which correspond to the users own perception of important image features. However, the information contained within an image is typically rich, subtle and enigmatic. It is open to subjective interpretation, contextual nuances and not always amenable to deterministic analysis. For these reasons it is not easy to compare one image against another. We take a closer look at the essential components of a practical IDBMS and concentrate on mechanisms which support automatic image retrieval by content. Such a system must make a provision for the following:

- Automatic feature extraction for constructing image indexes.
- Data structures to contain these indexes for multi-dimensional searches.
- Query methods to support image access.

This thesis aims to show how neural networks can be used to extract features for automatic indexing and, so that this work might be put in perspective, much of this chapter is given over to the current state-of-the-art in automatic image indexing. Data structures, storage methodologies and data management systems are mentioned only in passing.

We summarise the different approaches to feature extraction, the type of features that have been used index images to date and discuss the relative merits of each. At present the most successful methods have, on the whole, avoided the use of sophisticated image descriptors and use rather crude, low-level features or 'put a human in the loop' to extract objects from an image.

There is a general consensus that retrieval by content in IDBMS will push current image processing technologies to their limits and that neural networks hold much promise for this type of application. We discuss the role of neural networks in feature extraction for image retrieval by content systems.

The first tentative steps have been made toward tackling automatic indexing for IDBMS. It is, however, a very formidable problem. We present an analysis of the type of queries posed to a typical commercial picture archive and show that the subjective nature of these requests are not always amenable to conventional automatic processing.

We have a problem and we want a practical solution. We hope to show what that problem is, how current research trends attempt to tackle it and how neural networks can offer us a real alternative to conventional approaches.

2.2. The Growth of Image Database Technology.

Every day we are bombarded with information presented in the form of images. So important are images in the world of Information Technology that we generate literally millions of images every day, and this number keeps escalating with advances in imaging, visualisation, video and computing technologies. It would be impossible to cope with this explosion of image information unless these images were organised for rapid, reliable, flexible and convenient access on demand. The application spheres for IDBMS are widening as the cost of the technology to support it continues to drop. Such applications might include:

- Office and Library applications.
- Printing.
- Publication and Advertising.
- Security and Identification.
- Medicine.
- Geographic Information Systems.
- Education and Training.
- Fine Arts archiving.
- Entertainment and Broadcasting.

These communities have enthusiastically embraced multi-media technologies for further exploitation of their holdings*[1][2][3]*.

Methods used to retrieve images from these databases vary greatly from application to application. An art historian may want to retrieve images of a reclining model, a medical researcher may want chest images with a specified condition near the heart and an advertising layout editor may be looking for a picture he remembers of a beach scene with palm trees in it, for example. To be effective, such applications demand innovative approaches to a variety of issues and components including storage sub-system platforms, indexing and retrieval mechanisms and user interfaces.

Whilst the technologies which store and transmit image data have, by and large, developed at a steady rate, it is true to say that the methods to retrieve this data in a convenient and flexible way have not. The marked lag of automatic image indexing and retrieval mechanisms behind the development of the storage technologies has created a situation that can be likened to that of a library kept behind locked doors. The dichotomy between these two inter-dependent technologies has highlighted a very pressing need for the development of practical access mechanisms for image databases*[2][4]*.

2.3. Contrasting Conventional and Image Databases.

Computerised text database management systems came about because users desired convenient access mechanisms for archives of alpha-numeric type data (in the same way that they now want convenient access mechanisms for image databases). In these systems, large amounts of data are organised into fields and important or key fields are used to index the database making the search very efficient. These information management systems are limited by the fact that they work well only with numeric data and short, one-dimensional alpha-numeric strings. However, image data is, typically, richer in information content than simple alpha-numeric text; we have all heard that 'a picture paints a thousand words'. The radical difference between image and alpha-numeric data has meant that classical approaches to text database design cannot be readily applied to image databases.

Most current commercial IDBMS base their retrieval mechanisms on the use of keywords or text associated with each image and do not directly capture the visual properties of the data *[5][6][7]*. Queries are performed using standard query languages such as SQL. This allows keywords and text queries including logical combinations, conjunctions (AND's), disjunctions (OR's) and negations (NOT's) of image/text predicates.

There are several problems associated with these methods:

- The search is dependent solely on the key words. If the current query refers to image properties that were not initially described, the search will, in all probability, fail.
- Some visual properties are difficult or nearly impossible to describe with text such as certain textures or shapes. See, for example *[8]*.
- Even if all useful characteristics of an image are described with text, there is no commonly agreed-upon vocabulary for describing image properties, so that a 'curvy' item may not match a 'wavy' one, for example.
- Whereas text may be regarded as a unique data representation, multiple representations are possible for images. At present it is difficult to incorporate a mechanism that will deal adequately with such ambiguities in the framework of existing practices.
- The descriptors for the image indices must be selected from a code book of object/feature types. For a database whose contents are in a state of flux, the code book entities may have to be continually updated. This is not an easy task.

It is evident that a totally new approach to organisation, indexing and query processing is needed. We must consider the issues in visual information management rather than simply extending existing database technology to deal with images. IDBMS encompass not only databases but aspects of image processing and image understanding as well - this is shown below in fig. 2.1.

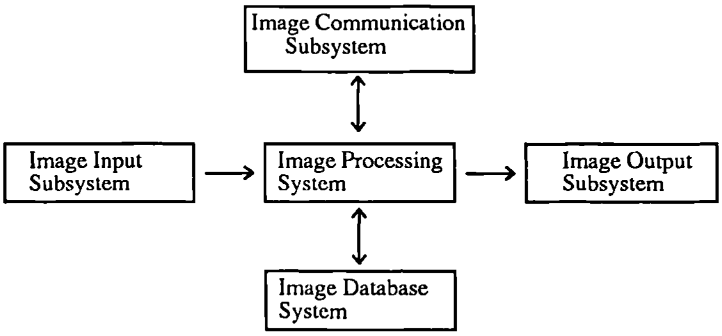


Fig. 2.1 A typical Image Information System.

2.4. Elements of a Content-Based Retrieval Mechanism.

Given a large image database, how can we retrieve images from it? There is an increasing body of research which is moving toward content-based retrieval methodologies. Here, retrieval is based on shape, colour, layout and position of objects in an image, or motion of objects in a video sequence, for example, and uses sample images or user sketches to specify a query. In this context the type of user queries that might be encountered are; "Find images with an object of this texture", "Find images with a significant amount of red", "Find objects which contain images similar to this one", for example.

In a conventional database system, the access mechanism is facilitated by an index which forms a 'hook' for queries posed by the user. In general, one can regard indexing as a means of reducing the search space of an operator without losing any relevant information. In the case of an image database, an index may be regarded as a mechanism for supporting the efficient search of a collection of images, based upon intrinsic properties of the images and their semantic content. There are a variety of types of features to index which might include entities/objects, attributes, relationships and derivations of these.

Most current Image Database Systems use manual descriptors for each image in the database [9]. Manual coding of these features is fraught with problems - not least of these is that each image must be manually indexed before being incorporated into the database. This is a very time-consuming and laborious exercise which makes this approach a rather unattractive prospect for most potential users. In an IDBMS it is essential to at least partially automate the data capture and indexing operations. Automated indexing requires automated (or operator assisted) feature extraction and (potentially semantic) interpretation which, in turn, requires a combination of an efficient visual query interface, pattern recognition primitives and robust similarity measures.

Content-based retrieval relies heavily upon computer vision and pattern recognition techniques to formulate efficient indexes. A typical Image Processing sub-system for a retrieval by content IDBMS is shown below in fig 2.2. This system attempts to extract image features and their spatial relationships (they might be roads on a map or tumours in a medical image, for example) to form an index which allows two images to be matched using a metric similar to that of a human.

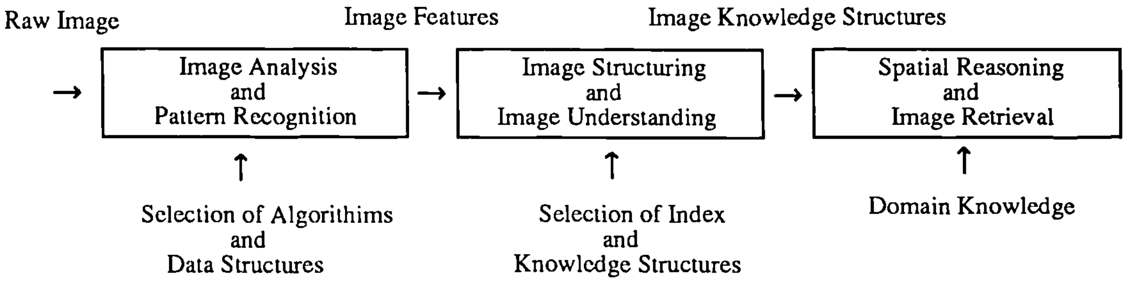

Raw Image
Image Features
Image Knowledge Structures
Image Analysis and Pattern Recognition
Image Structuring and Image Understanding
Spatial Reasoning and Image Retrieval
Selection of Algorithims and Data Structures
Selection of Index and Knowledge Structures
Domain Knowledge

Fig. 2.2 Operations performed by the Image Processing System.

The sophistication of the access mechanism in a IDBMS must match the sophistication of the data contained therein and applications involving image data demand an innovative approach to a variety of issues. For example, a comprehensive description of all of the image entities and their interrelationships would permit us to pose fairly sophisticated queries to the system. However there are several issues at stake here:

- The data models must accommodate complex image descriptors which invariably involves memory intensive hierarchies. This memory grows exponentially with the number of features used in the index. Comparing these models in the retrieval stage is often very computationally expensive. Can we afford this?

- Does the similarity measure support more complex image descriptions? Similarity measures frequently 'fall apart' as the dimensionality of the metric increases.

- How robust are the descriptors? Low-level (and comparatively simple) indices frequently out-perform more sophisticated ones *[10][11][12][13].*

The image database must be large to justify the use of the relatively expensive technology required by these methods. Small to medium databases, or even large ones with good text descriptors, can be manually viewed with a good fast browse of 'thumbnail' images (reduced versions of size, say 100 x 100, where 50 to 100 can be simultaneously displayed on a screen) and this may be sufficient in many cases .

The key issues to be addressed are: feature representation, similarity measures, data organisation and query languages.

2.5. Shape and Feature Representation in an Image Index.

The level at which features are represented can have a marked effect upon the performance of the index. Real world images typically contain noise and occluded objects within a scene and, ideally, the index would support a matching mechanism which could make a provision for this. The features can be represented at several levels of sophistication - these are discussed below.

2.5.1. Global and local features for indexing

The levels at which the image may be represented by the index range from full-scene descriptors *[14]* to elemental primitives *[15][16]*. Full-scene descriptors do not require that objects be isolated from within the image and do not suffer from some of the difficulties typically encountered during this phase. However, global features are far more susceptible to occlusion than local features and image recognition of occluded (or touching) objects is beset with problems *[17]*.

This can be circumvented by characterising images in terms of local rather than global features*[15]* - the more primitive the model, the more robust the representation. However, the data models are far more complex for local features and this, in turn, introduces other problems - see 2.8. Examples of local features are line and curve segments of the object boundary and points of maximal curvature change, for example *[18]*.

[16] uses primitive local features, such as edges, to construct domain specific models which make up the model-base. The model-base is compiled off-line with the aid of a domain expert who interactively refines the model objects. This scheme seems to work best for images which have a high degree of formality in their pictorial representation such as circuit diagrams and topological maps, for example.

2.5.2. Spatially dislocated features as indices.

In this scheme only the object information is extracted from the image - their interrelationships are ignored. A query supported by this might be 'Find all images that contain one or more objects present in the input image'. Once isolated from the background (either automatically or, as in *[10][19][20]*, interactively) the salient characteristics of the object must be fused within the index. Niblack *et al* define the objects in terms of area, circularity, eccentricity, major axis orientation and a set of algebraic moment invariants *[10]*. Note that this technique cannot deal with occluded objects.

2.5.3. Object/position descriptors as indices.

Here the spatial inter-relationships of the image entities are embodied within the index as ordered sets of shapes or graph structures. Such an index might support the query 'Find all images with this object in this position'. Finding the relations among objects has two problems: the spatial relations are not usually precisely defined and the number of such relations may become too large for a practical indexing scheme*[10]*.

The structuring of the data in such an index is dealt with more fully in 2.9.

2.5.4. Low level features as indices.

In many applications, low-level image properties such as colour *[13][11][12]*, texture*[11][21]*, and spatial derivative primitives *[16]* have a broad, intuitive applicability and can be very efficient as indices to an image database. These features do not suffer from the drawbacks associated with the previous two techniques and, recently, some fairly encouraging results have been reported. The success of these techniques can be attributed to the fact that there is no need for specific image objects to be identified within a scene. Considerable research activity has focused on ways to characterise and recover qualitative descriptions of such attributes from within images for indexing purposes.

The QBIC IDBMS, currently under development by IBM *[10][11]*, has used colour, texture and shape features as indices and reports that the most favourable results to date have been achieved using colour spectra histograms as the index. It must be said here that such a coding scheme produces a highly ambiguous representation of the image, and its efficacy is probably a result of its simplicity.

2.6. Easing the Burden of Object Recognition for Index Construction.

The root of the problems associated with content-based retrieval mechanisms lie with the inherent inadequacies of existing computer vision models. Object recognition, in particular, has proven to be a particularly challenging problem and a robust, non-domain specific solution has yet to surface. Isolating those features which correspond to the type of query metric that a human might employ is not easy. However, if the automatic object recognition phase of the indexing mechanism can be 'short circuited' then far more encouraging results can be achieved.

We have already discussed how low-level features and full-scene descriptors can be used to form image indices with the need for an object recognition phase.

Sometimes, for domain specific applications, we will have *a priori* knowledge of the objects present within all the images in the database. In such cases we can use a pre-compiled code-book of these icons, often called a model base, to derive the appropriate codes for the object entities within an image. Such models relieve the burden of object recognition and can improve the performance compared with more open-ended and less constrained systems*[15][16]*.

2.6.1. Interactive object recognition.

One alternative to fully-automatic object recognition is 'to put a human in the loop' and use manually selected object descriptors. Ordinarily, this is a very time consuming operation and simply not practical for large image data bases. To speed up this process, many researchers have proposed the use of interactive tools to help the user to select features for the index quickly and simply. Niblack*[10][19]* and Samandi*[20]* present an interactive method for outlining important image objects semi-automatically. The outlining mechanism is steered by user cues which are entered through a mouse.

A further advantage of this technique is that once an object is separated from the background it may be normalised so as to effect similarity retrieval invariant of size, scale and rotation artefacts.

2.6.2. Indexing CAD drawings

Within a CAD drawing most, if not all, of the image entities are derived from icons defined within a drawing package - these icons are usually represented as distinct codes which can be extracted comparatively easily from the drawing data. Thus, objects may be coded without the need for the recognition phase and some quite encouraging results have been achieved with circuit and plant layout drawings, for example *[22]*.

2.7. Similarity Measures for Image Retrieval Systems

This metric forms the basis of the retrieval decision mechanism and should facilitate a degree of fuzzy-matching across all images within the database. Obviously, this metric, which embodies both aspects of shape similarity and position similarity (if, indeed this is included in the index), should correspond to a human beings notion of similarity. This is not at all a straight-forward matter for real world images and is complicated by contextual sensitivity and subjective interpretations of the users visual perception. Mumford *[23][24]* has pointed that retrieval by shape similarity, for example, is a formidable problem.

Matching images for database retrieval systems requires a robust similarity measure. It is generally held that the matching of objects using similarity based on several features will be one of the most common and fundamental operations *[2]*. Techniques to judge similarity among different patterns will, to a large extent, depend upon the nature of the application. *[2]* concludes 'What is needed is the development of a general theory of similarity that will be useful across several applications'.

The number of similarity metrics increases with the number of features embodied within the index. Thus, the errors arising from the mismatch between a machines' and a humans' notion of similarity are compounded. As a consequence, many similarity measures 'fall-apart' as the dimensionality of the index increases. This probably accounts for the relatively superior performance of simple metrics over more complex ones *[10]*.

The robustness of the similarity measure is entirely dependent upon the space in which the comparison is undertaken. Each image is represented as a point in this space and the distance between the two points should correspond to a human beings notion of visual similarity. The object of the retrieval mechanism is to recover those images which are closest to the target point defined by the query. The following similarity measures have been used:

Index Root	**Distance Measure**
Colour*[13][10][12]*	Euclidean distance in weighted RGB space. Weights depend upon quantised colour spectrum of input image .
Texture*[10]*	Euclidean distance in weighted three-dimensional space of texture components. Weights depend upon the variances of each texture component .
Full Scene *[89]*	Euclidean distance in the global feature space learnt by a neural network.
Shape *[10]*	Weighted Euclidean feature space represented by the shape moments. *[10]* reports that the distances in this space do not correspond particularly well to a human beings perception.

Fig. 2.3 Similarity measures for different feature attributes.

2.8. Query Processing.

Once the set of features for objects and images has been computed for the indexes then queries may posed to the database. A query might be initiated by a user in an interactive session by specifying an object or set of object attributes and requesting images with objects 'like the query object'. For example, images can be requested that contain the object whose colour is similar to the colour of an indicated object, or to the colour selected from a colour-picker menu *[10]*.

The nature of the query type for an image database retrieval scheme depends upon the perspective of the application domain:

- A video editor might wish to access specific shots or scenes, based on a rough description or sketch.
- A Geographic Information System (GIS) user would pose complex special purpose queries to locate a particular shape that has the desired features etc.
- An image archiver requires rapid browsing and scanning (possibly by the use of user defined filters) regarding content and features, for example.

Most researchers in the field regard query-by-example as the most promising way forward for image data retrieval *[2]*. Queries of this sort may be initiated as sketches or interactively via a user menu.

2.8.1. Query by sketch and images

The schemes adopted by *[10][25][26]* filter out important line segments so that the normalised query image is reduced to a kind of binary cartoon-like outline drawing. This is sub-divided into a number of patches and each patch compared with the database image (which has been pre-processed with the same transformation) through a logical binary correlation operator. Kato*[26]* does not represent the processed images as indexes but performs the appropriate transformations at search time. This means that the search times can be very long for large databases.

Rickman*[27][28][89]* uses a neural network transformation to produce an index from a query image. The neural network learns the transform which preserves most information about each image within the database. This technique learns 'global' features which characterise the distribution of image types within the database - such 'full scene descriptors' avoid the need for the problematical object recognition phase but are sensitive to translation, scale and rotation artefacts.

2.8.2. Query by colour.

The colour key could be isolated from a query image or chosen via a user interface through a 'colour picker'*[10]*. However, matching images solely through their colour spectra can produce ambiguous results.

2.8.3. Query by subjective descriptions.

A users description of an image is highly subjective. Kato *[25][26]* presents an image coding scheme that can embody some of the users subjective interpretation into the query. He employs a mixture of text descriptors and pictorial information that are fused together to form the query. The text descriptors are chosen from a list of adjectives (such as 'cheerful', 'soft', 'elegant' etc.) and weighted accordingly.

However, entering the subjective features for each and every image in the database is laborious and the subjective impression of each image may vary from user to user. Kato points out that there is a degree of correlation between the subjective interpretation and the colour spectrum histogram of an image. Once the mapping between these two domains is known the subjective label can be assigned automatically by looking at the colour spectrum histogram. The relationship between the colour spectrum histogram and the user selected keywords (and their weightings) is found during a learning phase where the user is asked to ascribe keywords to a number of images. Multivariate analysis is used to calculate the mapping between the colour domain and the subjective description domain.

A similar technique is used to map graphics features (spatial frequency, local correlation and contrast, for example) to the users notion of similarity. Here the user is asked to group together a number of images into distinct clusters and the mapping between the subjective interpretation of similarity and the graphics features calculated through multivariate analysis. This method has been used successfully with both Art and Trade mark databases.

2.9. Data Representation and Organisation

The primary objective of image retrieval in an IDBMS is that it supports rapid selection of images fulfilling the query criteria. To this end it is imperative that the representational models which describe the image objects and their relationships be structured in a manner which permits indexing and similarity measures to be performed efficiently. Efficient data representation is the key to effective memory management in Visual Information Management Systems - high performance data representations and structures are essential. Examples of structures for image data are pyramids and quad-trees. In these data models the image is represented by an ordered, or partially ordered set of shapes or by a graph structure which reflects the spatial relationships of the objects. It has been shown that indices which support a hierarchy of feature descriptors enable more effective searches than non-hierarchical ones. Grosky *[15]* points out that models based on hierarchical features are both more adaptable and more efficient that models based on complete image objects. Generally speaking, specific representations of indices within these data structures support compression, retrieval of occluded and non-occluded objects based on shape and searches for contained and overlapping features.

Two prevailing data models are used in current systems: relational and object-oriented - in conventional data-base practice the latter is gradually gaining popularity over the former. This is because object-oriented management systems can nest various type constructors (arrays, lists, tuples) to any level which permits data structures to be defined with a richness that is not possible in relational models. For example, in an image many of the objects being modelled will, in turn, be parts of the representational data. This type of recursive data model can be represented rather neatly using an object oriented approach which might be used, say, to define image regions that are labelled as domain entities as attributes for other entities.

Each image within the database corresponds to a point in a multi-dimensional space. Since image queries are typically based on a large number of queries, the data structure which accommodates the image description should support multi-dimensional indexing and similarity matching. The main multi-dimensional indexing structures used at present are:

- R*-Trees *[29]* and R-Trees *[30][31]*.
- Linear Quad-trees *[32]*.
- Grid-Files *[33]*.

These data structures all been used successfully for text databases. However, most multi-dimensional indexing methods explode exponentially for high dimensionalities *[34]*. Whilst most of these structures may be efficient for text searches, they cannot be readily exploited for image queries which might have as many as 20 dimensions. Linear quad-trees, in particular, suffer from this and are not efficient as image indexes. Niblack *et al [10]* reports that R*-Trees seem to be the most robust for high dimensions.

Since data hierarchies are very sensitive to the number of dimensions employed for the indexing metric, the feature extraction method must be such that only a few features are sufficient to differentiate between objects.

In many of the application domains for IDBMS, data must be represented in multiple scales to cater to the needs of users requiring different levels of abstraction. If data is not explicitly represented in multi-scales, they need to be computed as and when they are requested. Such interpretations and data handling depend upon the semantics of the domain.

The richness and scope of information contained within an image means that several alternative interpretations are quite possible. Feature extraction, labelling, decisions made in processing, and assignments of values made by humans are all examples of interpretation. Generally, interpretation carries with it some amount of ambiguity, error, and uncertainty. To be effective, the data model should go some way toward supporting alternative interpretations of data, views, ambiguity and uncertainty within a database. This challenging issue is unlikely to be solved in the very near future *[2]*.

The index allows the features to be extracted from the image prior to its inclusion in the database so that the matching phase can take place in rapid fashion. The upshot of this is that the indexes must be instantiated before matching and the ranking of these representations and cannot take the context of each users preference into account. This problem is especially pressing in large and unconstrained collections of images.

Figure 2.4 outlines some typical IDBMS and illustrates the current research trends in this increasingly active field.

2.10. The role of Neural Networks in IDBMS.

Any efficient image retrieval scheme for an IDBMS must represent the images as indexes. These indexes must capture the salient features within each image and enable them to be compared in a rapid and flexible manner. However, extracting indexing features has proved to be an extremely difficult process to automate using conventional image processing techniques as those features deemed to be important to a human are often not amenable to deterministic analysis.

Neural networks have received much attention as information processing architectures and have been applied to a wide range of computer vision problems *[35]*. They consist of large arrays of comparatively simple processing elements, or 'neurons', which interact in parallel to learn a mapping from a problem to a solution in a non-deterministic manner. In this respect both the architectures and the learning mechanism resemble the processing topologies known to exist in biological systems.

Rickman*[27][28][89]* shows how an array of neural networks may be used to extract features which form the basis of an automatic indexing mechanism for an IDBMS. The neural network represents each image with respect to a number of features learnt during a training phase. The feature exists as a point in multi-dimensional feature space and the index represents the length of the vector between the image and this feature. The neural net transform (from image to index) preserves the relative relationships between the images in the original domain - the similarity between any two images is given by the

System Description	Query	Data Model	Index Method	Index Extraction	Data Structure	Applications	Ref
PICDMS: Picture DBMS using dynamic stacked images and gridded data	Command language ADD (IMAGE FIDD FIX (8,0) DIFF = BAND4 + BAND5	Stacked image	Field name with current location	pre-defined or manual	Flat file (3D matrix of stacked images)	Image processing	*36*
IIDMS: Intelligent image database system using 2-D string as iconic index	iconic query-by-pictorial-example by drawing pictorial query	Relational	string	Automatic or Manual.	sigma-tree	Image processing	*9*
Visual structure database	Symbolic query Car in front of house	entity-relationship diagram	attributes	manual	quad-tree and entity-relationship records	cartography	*37*
GRIM_DBMS: Automatic extraction of objects and semantics using pattern recognition and image processing with fuzzy match	Command languages RETRIEVE IMAGES (hospital/0.9) CONTAINING (double_bedroom/1.0)	attributed relational graphs	predefined attributes as cluster indexes	automatic	tree with cluster indexes	CAD/CAM	*38*
I-See: Software environment emphasises precompilation and query by image content	Iconic query as guide to search. Symbolic query SHOW cities WEST_OF city name = 'Pittsburg'	object oriented		automatic using image analysis and AI techniques		Image processing	*39*
IDB: Image Archiving by content	Match example image by similarity retrieval	object oriented	attributes	automatic using image analysis and AI techniques		Medical Image database	*40*
QVE/QBD	Query by visual example (graphic features and colour) and subjective descriptors.	relational	attributes and subjective descriptors mapped to attributes	automatic using image analysis and multi-variate analysis		Art Archiving. Trade mark databases.	*25* *26*
QBIC	Query by visual example (shape, colour and texture) and text descriptors	'starburst'	attributes	image analysis and interactive /semi-automatic image processing	R*-Tree	Pictorial Archiving	*10*

Fig. 2.4 Some typical IDBMS and their characteristics.

Euclidean distance between the indices. This neural net has been shown to provide a fast and fuzzy matching mechanism for a retrieval-by-content IDBMS.

Sanger *[41]* has shown that, when trained with a broad range of image types, such a neural network learns feature primitives which are almost identical to the 'retinal fields' found to exist within mammalian visual systems. These fields, located within the primary visual cortex, are used to encode images for subsequent processing stages 'higher up' in the cortex. This observation seems to validate the thrust of this approach. Indeed, other researchers have proposed the use of such fields for image coding systems *[16][42]* (though these implementations are not neural network based).

2.11. The Nature of Queries in a Typical Commercial Pictorial Archive.

There is a strong inter-dependence between the criteria used for selecting an appropriate index and the nature of the queries put to a practical IDBMS. We might, for example, have an indexing scheme based on texture that can match images with a 100% success rate. But if users are unable to frame their queries in terms of texture then the retrieval mechanism supported by the index scheme is as good as useless. At this juncture it would be useful to analyse the kind of queries that might be posed to a typical commercial pictorial archiving agency.

Enser [43], investigated 2,722 queries put to a commercial archive of over 10 million pictures (of assorted types) with a view to assessing the potential for both manual and automatic indexing schemes meeting the needs of prospective IDBMS users.

2.10.1. Characterisation of requests

Enser divides the queries into 4 categories according to the 'uniqueness' of the subject matter being requested. The concept of uniqueness allows a particular occurrence of a request for a visual representation to be differentiated from every other occurrence of the same entity type. The 'uniqueness' or 'non-uniqueness' of any subject matter is further refined by additional qualifiers supplied with the request. The categories are defined below in fig.2.5.

Category	Definition	Example	Proportion of Total Requests
1	Non-unique with no added refiners	Paddle Steamers	42%
2	Non-unique with added refiners	Shell Shock after WWI	27%
3	Unique with no added refiners	George VI	6%
4	Unique with added refiners	George VI Broadcasting	25%

Fig. 2. 5 Categorisation of query types for a commercial Image Archive.

A random walk through the list of request reveals the following queries:

Query	Category
3D-glasses	1
Assassination	1
Tightrope walkers	1
Wanted posters	1
Girlies in nice frocks	2
Milkman in the rubble	2
Boxing babies	2
Old preachers	2
Jarrow march	3
Eaton beagling pack	3
Dalai Lama	3
Norman Tebbit	3
MacMillan in sport preferably shooting	4
John Lennon - c.1988	4
Mao Tse Tung, Red Army 1936 - 1938	4
Hitler addressing crowds	4

Fig. 2.6 Typical queries for a commercial picture archive.

It is evident from the nature of these requests that pattern matching techniques are unlikely to arrive at an index that would support such free-form queries. Indeed, for many of the requests, even manually selected indices would not allow the browser to navigate freely through such an unconstrained database. This work suggests that most user queries are so application specific and subjective that to attempt automatic feature extraction using present technology would be folly.

Enser concludes that '... *the indexing of commercial photographic material is of low utility. However, for subject-specific collections, the application of indexing terms* [whether entered manually or derived automatically] *are a far more viable prospect.*'

Though this report is somewhat sobering for a researcher striving to automate the image indexing mechanism, it does shed some light on the type of queries that an indexing scheme should support. The growing research interest and investment into IDBMS would seem to counter the claims made by Enser. The number of research papers published in this area has been escalating very rapidly over the past few years and this clearly attests to the very real need for IDBMS in a world where rapid and flexible access to electronic images will become increasingly important.

2.12. Discussion

The cost of electronic data storage media continues to fall as the speed and sophistication of data transmission technologies continues to rise - IDBMS are now a viable prospect. However, the methods to access electronic images in a flexible and convenient way have not been developing at the same rate. This mismatch between these inter-dependent technologies has created a very pressing need for practical retrieval mechanisms for IDBMS.

IDBMS technology will only really come to the fore when efficient, fast and user friendly access mechanisms become available. It is generally agreed that such mechanisms are likely to incorporate some facility for retrieval by content which requires that the salient features must, somehow, be extracted from each and every image within the database to allow two images to be compared. This could be performed either manually or automatically. The former is laborious, time consuming and unlikely to provide an attractive solution to the broad spectrum of potential users. The latter requires image interpretation techniques which have yet to achieve the level of sophistication to make them a practical prospect. There is a general consensus that image retrieval by content will only become a practical prospect when general purpose automatic feature extraction is possible. Fortunately, work is currently underway which is making progress in this direction and we are now beginning to see the first image retrieval by content systems meet the market place *[44]*.

The most promising results for automatic indexing for retrieval-by-content systems to date have been achieved using colour, texture, shape and 'full-scene' feature types. None of these approaches have attempted to isolate specific object features or their spatial relationships automatically. Their success is due to the fact that the features used are relatively crude. As the indices which act as scene descriptors become more sophisticated (in an attempt to support richer user queries) then following problems begin to manifest:

- Current pattern recognition practices are not capable of supporting robust general purpose scene descriptors.
- Hierarchical index structures which support multi-level searches grow exponentially with the number of features. At present, such structures can only practically support about 20 features. This does not permit a particularly rich description of real world images.
- Similarity measures begin to fall apart for multi-dimensional indices.

Indexing could be performed manually, but, for a large database, this is simply not practical. In any event, retrieval from fairly small databases is probably best tackled by implementing efficient browsing mechanisms which allow the user to peruse sets of 'thumbnail images' efficiently rather than retrieval-by-content schemes.

Feature based indexing for retrieval of information is one of the main differences between IDBMS and conventional databases. The computational requirements for processing images and video will push existing computers to their limits. Parallel processing, pipeline architectures, neural networks and other similar approaches for feature extraction need to be studied to facilitate fast insertion and query processing in IDBMS. Some specific problems that need to be addressed here are:

- The use of pictorial objects as indices.
- Selection of appropriate low-level attributes as indices.
- High dimensional features for indexing.
- Query formulation.
- Dealing with uncertainty and ambiguity to support fuzzy queries.
- Applying measures for similarity.
- Determination of basic pictorial operations on features and images to generate new indices.
- Managing complex objects within index data hierarchies.

In 1992 the National Science Foundation of America made the following recommendations regarding the development of IDBMS technologies *[2]*:

'Researchers from image processing and computer vision, knowledge representation and knowledge-based systems and databases must work closely to develop Visual Information Management Systems. It is also believed that such systems should be developed in the context of applications that will be of immediate interest in industrial, medical or scientific areas. Without concrete applications and ambitious implementation projects, most of the important and difficult issues are likely to be ignored. Considering the inter-disciplinary nature of the research, it is strongly recommended that the NSF fund a few major research efforts in this area.'

Chapter 3

The Fundamentals of Image Coding using Neural Networks.

3.1. Introduction

In this chapter we aim to show how a simple neural network architecture and associated learning rule can be used as the basis of a very effective coding mechanism for our image retrieval system. We introduce the neural network as an adaptive information filter which can learn the important features within a database of images. Each image may be described with respect to this feature set allowing them to be represented in a very low dimensional bound to facilitate fast and fuzzy pattern matching for our retrieval mechanism.

We give a potted overview of the fundamental issues in the development of neural networks. We highlight some of the factors pertinent to our coding scheme and present an appropriate architecture and learning law for our type of application. We indicate that the coding transform required by our scheme has much in common with a classical statistical analysis technique called Principal Component Analysis (PCA). However, PCA is computationally expensive and not well suited to high dimensional data such as a large database of images, for example.

We show how a learning law, found to exist in 'real-life' neurons can be developed and implemented on a single layer feed-forward linear neural network to produce a very efficient approximation of PCA. This type of learning paradigm is particularly suitable for our image retrieval mechanism. Much of the mathematical formalisms, insights and theoretical foundations presented in this chapter are applied in a practical context in chapter 4, which, effectively, forms a 'sister' chapter to this one.

3.2. Neural Network Research - Some Key Issues.

3.2.1. The 'curse of dimensionality' and the blessing of the neuron.

The real world is a jungle of information in which the sheer scale of the stimuli is dizzying and its scope vast. If sense is to be made of this glut of data we must compromise and take in only that information deemed important for the task at hand and process it as efficiently as possible.

Fortunately, such stimuli are highly correlated and contain much redundant data which may be discarded with no loss of information but with a huge increase in processing power. In a living animal it is the brain which performs this filtering process, sifting out needless redundancy to free up information pathways.

In an attempt to emulate some of these desirable filtering and data processing properties, computer scientists began to take a closer look at the micro structure of the brain. What they found, especially in regard to the contrast between processing topologies found in the brain and a standard serial computer, has largely paved the way and motivated the current enthusiasm for so-called neural network research.

3.2.2. The evolution of the species.

In 1943 McCulloch and Pitts *[45]* introduced a simple electrical model of a basic neuron which, they proved, was capable of performing any computation that a standard digital computer could perform. This model is shown below in fig. 3.1

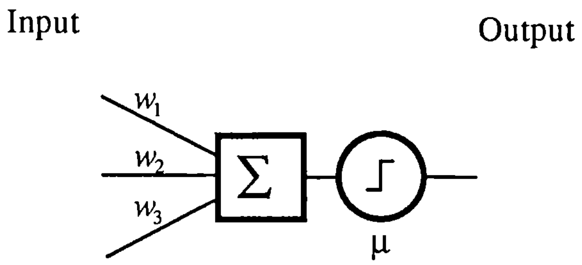


Fig. 3.1. A McCulloch and Pitts neuron

The model computes a weighted sum of its inputs and outputs a '1' or '0' according to whether this sum is above or below a certain threshold μ.

In 1962 Rosenblatt *[46]*, drawing on the McCulloch and Pitts theme, developed the perceptron and associated learning rule which enabled the weights to develop iteratively so that the desired computation could be performed (provided that what you desired lay within the bounds of what was possible with a perceptron; more about this later).

We may regard the perceptron as a sort of adaptive filtering network which, through a learning process, can attune to input the channels which are important for the neural calculation. It is this learning process which gives neural networks their remarkable ability to adapt to the nature of the problem at hand in a non-deterministic fashion. This is where their promise lies.

In the case of the perceptron, the stimulus for learning comes from a teacher where the output of the node is compared against the known answer. This is called supervised learning. Here the neuron is trained with a representative sub-set of problem/solution pairs and, hopefully, if the training set, architecture, learning paradigm and network functionality are correct, the net will be able to generalise and solve problems not yet encountered. This mixed bag of parameters has kept neural network researchers busy for a long time and probably will for a long time to come.

A trained neural network will perform a transformation on its input: the precise nature of this transform changes iteratively (and, with luck, converges) during the training process and is dependent upon the nature of the problem domain and the functionality of the node. To understand how learning influences the nature of this transformation in the perceptron it is useful to consider the following simple case: We have a set of P N-dimensional patterns in the pattern domain, $(X_1, X_2, \ldots\ldots, X_p)_,$ where $X_1 = (x_{11}, x_{12}, \ldots\ldots, x_{1N})$ etc. In this example the data-set consists of two distinct pattern classes and the role of the neural network is to develop a nodal transformation that will filter one class from another unambiguously. To this end we might require that the output of the node to go to '-1' for one class and '+1' for the other. These desired responses $(\zeta_1, \zeta_2, \ldots\ldots, \zeta_P)$, one for each pattern in the training set, are supplied by the teacher. The objective of the training is to correlate the input pattern with the desired response.

During the training phase the weights are adapted according to the discrepancy between the actual and desired output from the neuron in an attempt to bring the former in line with the latter. Here the weight vector $W = (w_1, w_2, \ldots. w_N)$ forms an N-dimensional hyper-plane which, we hope, will separate the two classes in pattern space. This is shown below in fig. 3.2 for N=2. The binary threshold unit activates such that any pattern one side of this plane elicits a '+1' from the node and a '-1' if it lies on the other. This hyper-plane is often referred to as a decision plane.

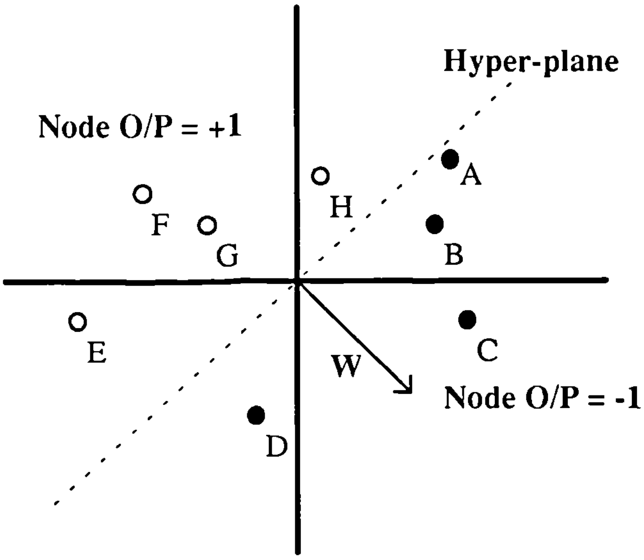


Fig. 3.2. Weight vector forms a hyper-plane which separates classes in pattern space.

The output of the node is given by:

$$O_i = sgn\left(\sum_{j=1}^{N} x_{ij} w_j\right) \qquad (1$$

During training the decision-plane is moved in the pattern space according to the discrepancy between the actual output O_i and the desired output ζ_i. The weights are updated as:

$$\hat{w}_j = w_j + \Delta w_{ij} \qquad (2$$

Where:

$\hat{w}_j$ is the new value of the j_{th} weight after training on the i_{th} pattern.

x_{ij} is the j_{th} element of the i_{th} training pattern.

$$\Delta w_{ij} = \alpha(\zeta_i - O_i)x_{ij} \qquad (3$$

Rosenblatt ensures that a degree of confidence is attributed to the positioning of this decision plane by including a margin which guarantees that the points lie at least a fixed distance from the decision-plane. The learning law now becomes:

$$\Delta w_{ij} = \alpha\Theta(Nk - \zeta_i y_i)\zeta_i x_{ij} \qquad (4$$

Where:

Θ is the unit step function.
N is the size of the input vector.
k is the margin size.

$$y_i = \sum_{j=1}^{N} x_{ij} w_j \qquad (5$$

This is the perceptron learning rule and has been proved to converge by Minsky[47].

3.2.3. A simple coding scheme using neural networks.

We have shown how a simple perceptron can learn to filter a data-set containing two distinct, linearly separable data-types, through a supervised learning scheme. Let us have a closer look now at how a perceptron might be used as the basis of a coding mechanism for our image database retrieval system. The basic architecture is shown below in fig.3.3.

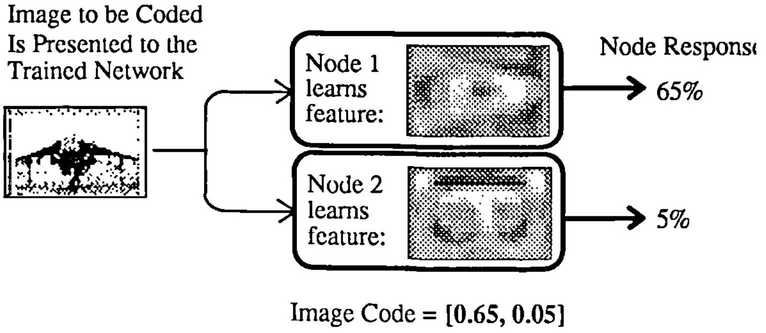


Fig. 3.3. A simple coding architecture using neural networks.

The output of the node forms one element of the code. In this simple example each pattern is represented by a 2 element vector. The objective of the training is to adapt the weights to demarcate the patterns as effectively as possible, rather like the scheme shown in fig. 3.2. The code should support rapid fuzzy matching of images across the database.

An appropriate Neural Network Transformation (NNT), from image at the input to code at the output, should preserve the relative relationships of the patterns in the original domain.

Let us suppose that the contents of the database consist of a number of distinct distributions of images and the objective of the training is to position the hyper-plane to demarcate them as effectively as possible, rather like the scheme shown in fig. 3.2.

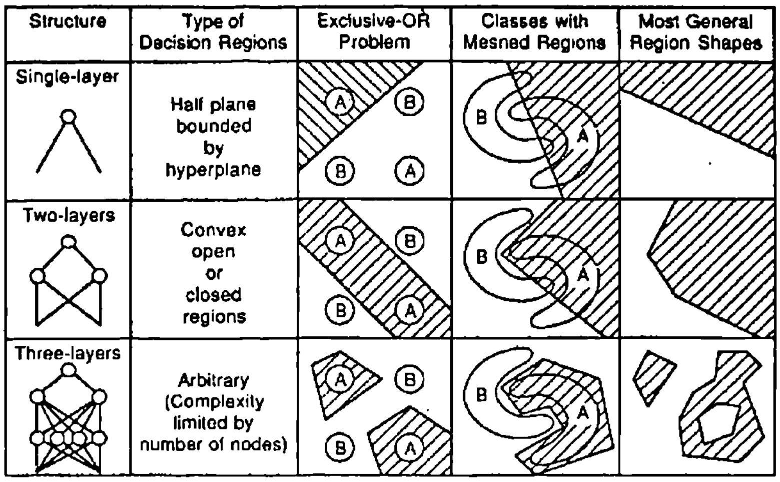
Since the perceptron is a Binary Threshold Unit (in that its output is either 'on' or 'off') it will produce a one-of-N code (in this case N=2) whose contents are dependent upon which side of the hyper-plane the pattern is lying. Whilst this code will provide excellent inter-class distinction it does not furnish us with much information regarding the extent of similarity between images in the database. If the code is to allow us to retrieve images based on a measure of similarity, then a simple binary threshold device such as perceptron will not suffice. This argument is extended in the context of Logical Neural Networks in 5.4.3.

A more appropriate NNT would be one that reflected the distance from the hyper plane so that the node output preserved the relative relationships in the original pattern domain. This is frequently referred to as a conformal mapping. In any event, it is highly unlikely that our database will contain notionally distinct classes of data required for supervised learning. This raises an interesting issue and one worthy of a more detailed investigation.

3.2.4. What perceptrons cannot do.

The perceptron learning rule is good for the data shown in fig.3.2 which can be partitioned effectively by a hyper-plane. Unfortunately, many problems arise where this is simply not the case as Minsky and Papert pointed out in their damming book 'Perceptrons' *[47]*. They showed that a single-layer perceptron could not even perform the, by now infamous, Exclusive-OR function. Rosenblatt retorted that multi-layer perceptrons had the potential to form arbitrary decision boundaries which could circumvent the problem of linear inseparability between classes in the pattern space. This is shown below in fig. 3.4. However, the perceptron learning rule cannot be applied to multi-layer configurations as it does not make a provision for updating the weights in the intermediate layers.

Enthusiasm and research funding waned and neural network research went into hibernation for about 20 years.



Structure	Type of Decision Regions	Exclusive-OR Problem	Classes with Mesnad Regions	Most General Region Shapes
Single-layer	Half plane bounded by hyperplane			
Two-layers	Convex open or closed regions			
Three-layers	Arbitrary (Complexity limited by number of nodes)			

Fig. 3.4. Relationship between decision regions and network topology for perceptron architectures.

During this time a few die-hard researchers were quietly working on this problem and independently hit upon a method which allowed the weights in the hidden layers to be updated. This method, generally referred to as backpropogation, heralded something of a Renaissance for neural network research which we are still enjoying today.

Backpropogation may be viewed as a generalisation of the perceptron learning rule for multi-layer non-linear neurons. This multi-layer architecture and associated learning law addresses many of the objections raised by Minsky and Papert*[47]* regarding the perceptrons' inability to cope with problems which are not linearly separable. Like the perceptron learning rule, backpropogation adjusts the weights of the network in an attempt to reduce the errors at the output of the node.

3.2.5. Backpropogation for image coding

Our own requirement may be framed in the context of an encoder problem where a high dimensional input vector is to be described by a low dimensional output vector. This type of problem is something of a benchmark test for the backpropogation learning algorithm and has been studied extensively*[48]*. One particularly interesting architecture pertinent to our application has been proposed by Cottrell *et al [49]*. Their model is used for image compression which is effectively what we are attempting to do with our coding process, the difference being that our code will not be used to reconstruct the images, merely to compare them in a lower dimensional bound.

Their architecture is a three-layer affair where the input and output layers have the same dimensionality. This is shown below in fig. 3.5. The network is trained as an auto-associator where the input and output nodes are clamped to the pattern presented to the net and the weights adjusted to minimise the output error using the standard backpropogation algorithm *[48]*. Because the training is auto-associative it requires no teaching input (unlike standard backpropogation) - this is often referred to as Self-Supervised training.

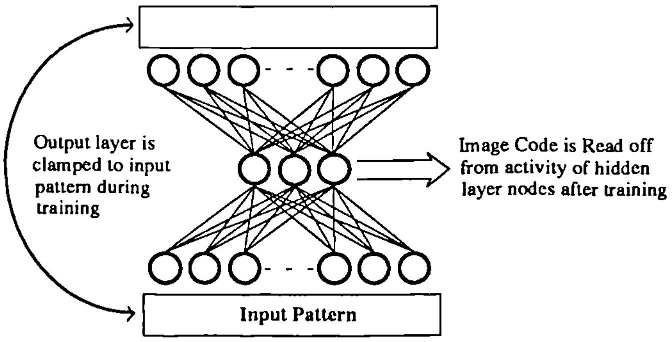

Output layer is clamped to input pattern during training
Image Code is Read off from activity of hidden layer nodes after training
Input Pattern

Fig. 3.5. A neural network architecture for image coding using backpropogation.

The input pattern passes through an information bottle-neck because it must be described by a relatively small number of hidden units and the reconstruction on the output layer is based on the activity of these units. The output function of these units is a simple sigmoid which limits the dynamic range of their activity. This so-called squashing function also introduces some non-linearity into the NNT.

Cottrell shows that this technique is quite capable of producing a code with a performance similar to that achieved by a classical statistical information filtering technique called Principal Component Analysis (PCA). However, backpropogation is typically rather slow and this application is no exception. Learning in backpropogation is motivated by gradient decent - that is, the problem is defined in terms of an energy landscape, the lowest point of which represents the optimal solution. The iterative solution manifests as the system gradually descends into a lower energy state during training. Baldi and Hornick *[50]* show that the training is very slow because this energy landscape has many plateaux on it. Cottrell suggests that it would be more practical to calculate the Principal Components directly using conventional techniques than to use Backpropogation to come up with the same solution. In conclusion then, whilst this technique is useful in that it sheds some light on the internal processes occurring within a neural network, it really does not provide us with a practical solution for image coding.

3.3. PCA and its Relationship to Our Coding Scheme.

PCA is a common method for analysing statistical data. In communications theory it is known as the Karhunen-Loeve transform and has been applied to the coding and characterisation of facial images by Kirby and Sirovich*[51]* and Turk and Pentland*[52]*. PCA has also been studied extensively in the context of pattern recognition by Oja*[53][54][55]*. PCA is also closely related to least squares methods in data clustering, factor-analysis, singular value decomposition and matched filtering. Linsker*[56]* notes that performing Principal Component Analysis is equivalent to maximising the information content of the output signal of a neuron in situations where the data has a Gaussian distribution.

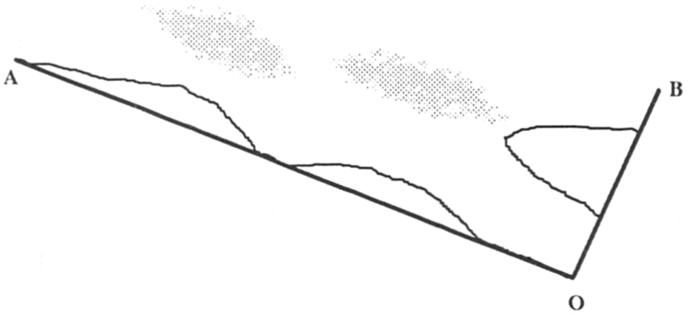


Fig 3.6. PCA transforms original data to a set of orthogonal bases which maximises the variance of a Gaussian data set.

The linear transformation produced by PCA effectively defines a new set of basis vectors where the projection of the data on these new axes maximises the variance of data set. This can help to sift out extraneous data to ease the burden of subsequent processing operations within a more manageable domain than the original. Fig.3.6 shows that such a transformation can highlight salient characteristics within the data. The resemblance between the transform produced by PCA and that required of our Neural Network Transformation is very striking.

The first Principal Component produces a linear transformation which rotates the axes so that the variance of the projection of the data points onto this axis is maximised. The second Principal Component is constrained to be orthogonal to the first and the transformation again maximises the variance of the projection subject to this constraint. The third Principal Components produces a similar transformation orthogonal to both the first and the second and so on. Generally speaking, successive Principal Components capture progressively less important artefacts within the data with the first Principal Component having the highest variance, the second the next highest and so on.

For a data set with a finite range of characteristics, dimensionality reduction is achieved by virtue of the fact that relatively few Principal Components are needed for a faithful description of the data.

The Principal Components are found from successive eigenvectors of the autocorrelation matrix of the complete data set. This is very a computationally expensive calculation and simply not feasible for anything but very simple data. It is certainly not a practical prospect for our coding scheme.

3.4. Unsupervised Learning in Neural Networks

3.4.1. What features should our coding network learn ?

The form of feature archetypes learnt during the training phase can have a marked effect upon the performance of the code. How is this?

Consider the image universe as represented by the contents of the database. Because of the nature of this database several clusters of data will tend to form in the universe (though the precise nature of such clusters may be quite arcane to an outside observer). Where the clusters are dense, that is, where there are a large number of images with similar features, we want the selectivity of the matching mechanism to be high so that we might distinguish between a large number of images. However, high selectivity in dense clusters may not afford us with sufficient generality to distinguish between images where the universe is sparsely populated. These two factors must be resolved if the code is to produce an efficient matching metric. Additionally, as we have already stated, the node should also preserve the relative distance measures between the patterns in the original space. The degree of similarity between the features learnt by the neural net (which, in the case of the perceptron model, form a hyper-plane) and the image presented to it, may be regarded as the length of the vector between them in pattern space. If the neural network has the appropriate functionality and is properly trained then the output can be made to give an indication of the length of this vector. These vector lengths form the basis of the code for that image. This is shown in fig. 3.7.

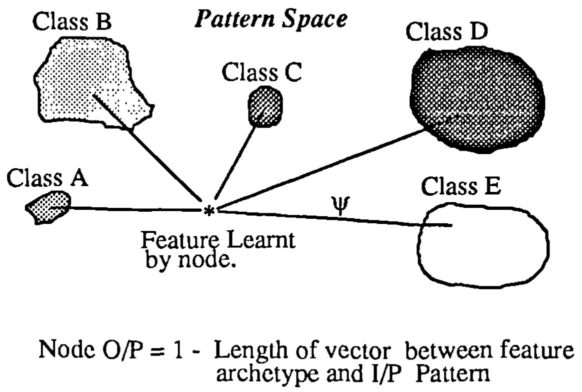


Fig. 3.7. Length of vector between learnt feature and image forms basis of code.

To maximise the selectivity of the code we should maximise the variance of the length ψ of this vector for the complete data set. That is:

$$\sum_{i=1}^{AllPatterns} (\psi_i - \overline{\psi})^2 = Max. \qquad (6$$

Selecting the training data to achieve this will enable us to encode the greatest number of images with optimum selectivity. The reader will recognise this as the thrust of PCA. Given that the nature of the pattern space occupied by the database is not well suited to supervised learning, we must now turn our attention to a form of learning which does not require a teaching input. This is termed unsupervised learning.

3.4.2. Hebb's law - a biological paradigm for unsupervised learning.

A common objective of any pattern recognition scheme is to determine good representations for data and remove redundancy. In the absence of a teacher, an unsupervised system must promote learning where there is a large amount of information within the data set and discourage learning where there is a high level of redundancy.

A good example of how biological mechanisms extract correlations from the input data in an 'unsupervised' fashion is to be found in the well-known Pavlov's Dog experiment. In this experiment, summarised below in fig. 3.8, the dog learns to associate the bell with the food and, after a short time, will begin to salivate when the bell rings.

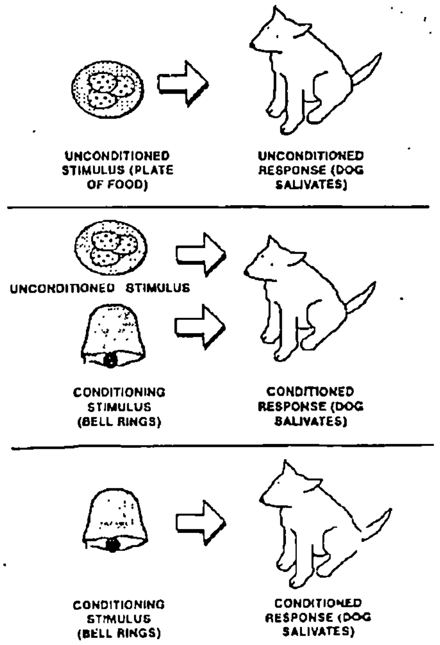

UNCONDITIONED STIMULUS (PLATE OF FOOD)
UNCONDITIONED RESPONSE (DOG SALIVATES)
UNCONDITIONED STIMULUS
CONDITIONING STIMULUS (BELL RINGS)
CONDITIONED RESPONSE (DOG SALIVATES)
CONDITIONING STIMULUS (BELL RINGS)
CONDITIONED RESPONSE (DOG SALIVATES)

Fig. 3.8 Pavlov's dog and unsupervised learning.

In 1949 Hebb *[57]* formalised this learning paradigm (though not mathematically) in a way which allowed it to be implemented on a neural network architecture. This learning law causes the weights to change in response to events which occur simultaneously within a node, effectively correlating the input and output activity. In the case of the 'Pavlov's dog' experiment, the relationship between the bell and the food stimulus are strengthened so as to induce salivation when only the bell stimulus is present. In a neural network, the connection strengths are modulated according to the degree of correlation between the input and the output. Referring to the perceptron model outlined in 3.2.2 , Hebb's learning rule becomes:

$$\Delta w_{ij} \propto x_{ij} y_i \qquad (7$$

Notice the similarity between this and the perceptron learning rule given in equation *(3*; the difference being that the former does not require a teaching input whilst the latter does.

For a single linear node the learning rule becomes:

$$\Delta w_{ij} = \alpha x_{ij} y_i \qquad (8$$

Where:

α is the Learning Rate

$$y_i = \sum_{j=1}^{N} x_{ij} w_j \qquad (9$$

For an N dimensional pattern.

If certain aspects of the data set occur with significant frequency then the system begins to learn these with increasing conviction and the system behaves as a data correlator for the entire data set.

3.4.3. Learning to forget

One of the shortcomings of Hebbian type learning is that if the learning process is allowed to continue unchecked, the node will eventually saturate so that every feature within the data set is embodied into the model solution. Such a node provides no discriminatory information for our coding scheme. To prevent this, some constraint must be introduced to limit the learning process. This limiting is usually achieved by regarding the learning capacity of the system as a fixed resource so that learning certain important aspects of the data set causes other, less important, aspects to be discarded or 'forgotten'. Hence, all Hebbian type learning rules have the following form:

$$\Delta w_{ij} = \underbrace{\Phi\left[x_i, y_i, w_j\right]}_{\text{Learning Term}} + \underbrace{\Psi\left[x_i, y_i, w_j\right]}_{\text{Forgetting Term}} \qquad (10$$

Since all learning is manifest in the weights of the neural network, it follows that in order to limit the learning resource we must impose a fixed constraint upon the weights.

3.4.4. Constrained Hebbian learning - a generalised approach.

Oja *[54]* indicates how we may derive a generalised learning rule given a learning rule and a constraint function. Consider first the unconstrained update rule:

$$\hat{w}_{t+1,j} = w_{t,j} + \alpha x_{i,j} y_i \qquad (11$$

Where:

$w_{t,j}$ is the j_{th} weight the prior to the update.
$\hat{w}_{t+1,j}$ is the *unconstrained* weight vector after the update.

Following the update, the weights must now be normalised to satisfy the constraint:

$$w_{t+1,j} = \frac{w_{t,j} + \alpha x_{i,j} y_i}{\Psi\left[w_{t,1} + \alpha x_{i1} y_i, w_{t,2} + \alpha x_{i2} y_i, \ldots\ldots, w_{t,N} + \alpha x_{iN} y_i\right]} \qquad (12$$

Where Ψ is a function which normalises the weights so that the constraint is satisfied. The value of this function can be estimated after the complete update by using a Taylor series expansion at α=0. That is:

$$\Psi(\alpha) = \Psi(_{\alpha=0}) + \alpha \frac{\delta\Psi}{\delta\alpha}\bigg|_{\alpha=0} + \alpha^2 \frac{\delta^2\Psi}{\delta\Psi^2}\bigg|_{\alpha=0} \ \ldots\ldots\ldots \qquad (13$$

If α is small we can ignore terms in α^2 and above which gives:

$$w_{t+1,j} = \frac{w_{t,j} + \alpha x_{i,j} y_i}{\Psi\left[w_{t,1}, w_{t,2}, \ldots\ldots, w_{t,N}\right] + \alpha \frac{\delta\Psi}{\delta\alpha}\Big|_{\alpha=0}} \qquad (14$$

Since we know that prior to the update the constraint must be satisfied this becomes:

$$w_{t+1,j} = \frac{w_{t,j} + \alpha x_{ij} y_i}{1 + \alpha \frac{\delta\Psi}{\delta\alpha}\Big|_{\alpha=0}} \qquad (15$$

Re-arranging and making the approximation that $\alpha w_{t+1,j} \frac{\delta\Psi}{\delta\alpha} \approx \alpha w_{t,j} \frac{\delta\Psi}{\delta\alpha}$ yields:

$$w_{t+1,j} = w_{t,j} + \alpha x_{i,j} y_i - \alpha w_{t,j} \frac{\delta\Psi}{\delta\alpha} \qquad (16$$

If we impose the constraint $\sqrt{\sum_{j=1}^{N} w_{t,j}} = 1$ then $\frac{\delta\Psi}{\delta\alpha} = y_t^2$ and the learning law becomes:

$$w_{t+1,j} = w_{t,j} + \alpha x_{i,j} y_i - \alpha y_i^2 w_{t,j} \qquad (17$$

or, more generally:

$$w_{t+1,j} = w_{t,j} + \alpha x_{t,j} y_t - \beta y_t^2 w_{t,j} \qquad (18$$

Where:

α is the learning rate.

β is the forgetting rate.

The learning law is motivated by the interplay between the level of correspondence between image artefacts and their differences - this is the upshot of the constrained learning resource and the action of the 'forgetting' term in equation *(18*. The act of learning a feature with a high correspondence to the archetype already learnt by the node increases the propensity of the system to learn a maximally similar or dissimilar one.

Since learning occurs most strongly when there are large positive and negative responses we can see that such a scheme will tend to increase the variance of the node.

3.4.5. The relationship between constrained Hebbian learning and Principal Component Analysis.

This learning law has been studied extensively by Oja *[54][55]*, Sanger*[41]*, Linsker*[56]* and others. This has revealed some very interesting properties:

i. Weights have been proven to converge to the Principal Component of the Training data with probability 1 *[54]*.

ii. After convergence the variance of the node output for the training set is maximised. Linsker *[56]* has shown that for zero mean data with a Gaussian distribution the node preserves the maximum amount of information within the data set. This is highly desirable for our coding scheme. In fact, this is just a re-interpretation of PCA.

iii. The learning rule requires only local data for the update; that is, no knowledge of the response to the other patterns need be stored. This makes it economical both in terms of memory and computational complexity.

3.4.6. Implementing a multiple Principal Component system on a neural network.

This learning paradigm produces a transformation which is directly equivalent to Principal Component Analysis and successive nodes should learn successive Principal Components of the data set.

Sanger *[41]* suggests a method which allows successive Principal Components to be calculated on a single layer, multi-node architecture.

The learning rule for a single node system is given in equation *(19*. To induce the system to learn orthogonal features in successive nodes, the weighted output y_i of unit i is subtracted from the input before it reaches unit i+1:

$$\Delta w_{ij} = \alpha x'_{ij} y_i - \beta {y'_i}^2 w_{ij} \qquad (19$$

Where:

Δw_{ij} is the change on the j_{th} weight of the i_{th} node after the learning update.

$$x'_{ij} = \hat{x}_{ij} - \sum_{i=1}^{1<j} y'_j w_{ij} \qquad \text{Orthogonalises inputs to successive nodes} \qquad (20$$

$$\hat{x}_{ij} = x_{ij} - \frac{1}{P}\sum_{k=1}^{P} x_{kj} \qquad \text{Ensures that the inputs have zero mean} \qquad (21$$

$$y'_j = \sum_{i=1}^{i=N} x'_{ij} w_{ij} \qquad (22$$

The architecture to support this learning law is shown in fig. 3.9.

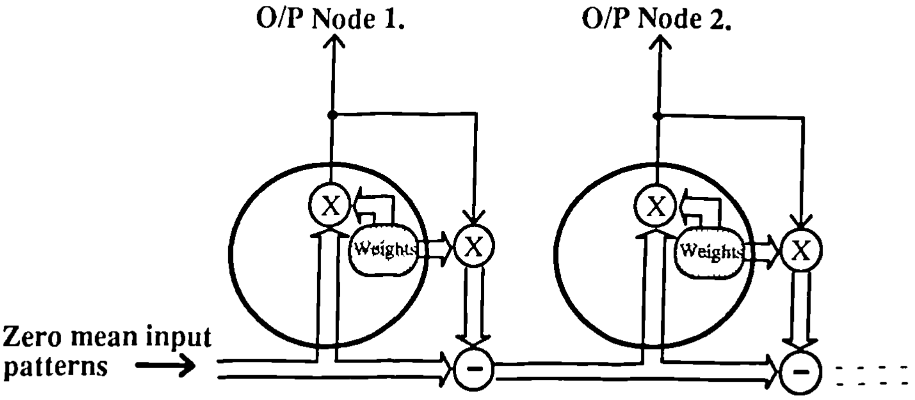


Fig. 3.9. Network architecture to support Sanger's learning rule.

Sanger proves that this technique provides a transformation which is directly equivalent to (though only an approximation of) PCA. The rule is particularly interesting as the nodes converge in parallel so that successive nodes can work towards a solution without requiring convergence in earlier nodes - though later nodes cannot converge before earlier ones.

3.5. Non-Linearities and Multiple-layer Networks - Are they Appropriate?

Bourland and Kamp *[58]* investigated the effects of non-linearities in neural networks performing dimensionality reduction with specific reference to the architecture proposed by Cottrel *et al [49]* and discussed in 3.2.5. They laid down a theoretical framework which showed that non-linearities in these nodes are useless and that 'optimal parameters can be derived from purely linear techniques'. They conclude that '... in general, training a non-linear unsupervised network to approximate non-linear functions is very difficult'. This is because the non-linear function space is very much larger than its linear counterpart so that fitting a function to data in this domain is very much harder than the linear case. We varify this with some practical tests and discuss this in more depth in chapter 4.

3.5.1. Is one layer enough ?

Since the advent of backpropogation, which allowed multi-layered neural networks to be trained successfully, many neural network researchers have regarded single layer nets as rather trivial structures whose performance could invariably be improved with the addition of a few 'hidden layers' of nodes. In fact, hidden layers will only provide us with arbitrarily complex, and presumably more appropriate transforms (see fig. 3.4), if the nodes themselves are non-linear. This is because a multi-layer linear topology will still, at the end of the day, only produce a linear function - a product of linear transformations is a linear transformation. In chapter 4 we provide experimental evidence which suggests that a linear function is adequate for our system. We may conclude that there is nothing to be gained from the addition of extra layers of nodes. This is rather gratifying as training multi-layer architectures is typically a complex and protracted affair.

3.6. Discussion

A neural network is basically an adaptive information filter which can be used to extract the salient characteristics of a data set during a training phase. We show how such a device can be used to transform an ensemble of images into a very low dimensional bound which enables then to be compared in a more manageable domain than the original. This Neural Network Transformation provides the basis of our coding mechanism.

For a device with such an enigmatic reputation, the basic architecture of a conventional neural network is remarkably simple: the output of the node is just the weighted sum of its inputs. The weight determines the importance that an input has in defining the solution to a particular problem with respect to the complete data set. The weights are adapted iteratively in a training phase during which the entire data set is presented to the node. These weights effectively form a hyper-plane which demarcates important regions within the pattern space; the relationship between the input pattern and this hyper-plane is given by the output of the node.

Our matching metric requires that images close together in the pattern space yield correspondingly similar codes in the much smaller code space. We point out that this criteria is not satisfied by a neural network trained as a classifier, such as the binary threshold perceptron. However, a 3-layered perceptron, configured as an auto-associator and trained using backpropogation can produce a dimension reducing transform very similar to Principal Component Analysis. This is, in essence, the coding transform that we require but excessive training times render this method as impractical as a classical implantation of PCA for large images.

In an attempt to circumvent some of these practical difficulties, we investigate a class of unsupervised learning paradigms found to exist in 'real-life' neurons called Hebbian Learning. This learning law adapts the weights to correlate the input and output activity of the node and is shown to be produce a transformation which is directly equivalent to, though only an approximation of, PCA. A neural network which approximates PCA will provide an optimal transform to preserve the maximum amount of information within the data set if the data has a Gaussian distribution within the pattern space. The approximations inherent in the derivation of the learning law means that the learning rate can have a profound effect upon the performance of the code. If the learning rate is small, the approximation is accurate, the weights will converge to the Principal Components of the data set but will take a long time to get there. If the rate is high, convergence will be quicker but the approximation is less accurate and the resultant transformation may not be optimal. These issues are taken up in chapter 4.

Chapter 4

An Empirical Study of the System Parameters for a Sum-of-weights type Neural Network.

4.1. Introduction

In this chapter we investigate the factors effecting the performance of a practical retrieval-by-image-content system. Initially, we work with a database of machine printed fonts which permits an objective appraisal of the performance of a neural network based scheme. Extensive practical tests and computer simulations are used to corroborate the arguments presented throughout both this chapter and chapter 3.

Much of the chapter is given over to the investigation of the learning rule developed in Chapter 3. The rate at which the neuron learns can have a profound effect upon the performance of the system. We investigate how the order of presentation of training data, its size and the size of the data set affect the choice of the learning rate. The performance is also dependent upon the diversity of images within the database - we investigate the extent of this relationship. We show how the code length and nodal transformation may be optimised with respect to the contents of the database. The nodal transformation is changed by re-training the system. This is, typically, a lengthy process and not always strictly necessary, dependent upon the extent of the novelty of the new images. We take a close look at the factors affecting the need for re-training.

This system works well; but is it optimal? We provide some theoretical and empirical evidence to show that, though the system is extraordinarily simple, further enhancements are unlikely to improve upon its performance.

We demonstrate how the coding scheme developed here may be used as the basis of an 'intelligent' video editing system.

4.2. An Image Coding Scheme using Constrained Hebbian Learning

4.2.1. Does this strategy produce a conformal mapping?

A single neuron trained using the learning strategy presented in 3.4.6 can be used to transform a data set into a more compact and manageable dimensional bound than the original whilst still preserving its salient characteristics. Consider the set of 16x24 bit binary characters, shown in fig. 4.1, in which an 'A' font is gradually, and evenly, merged to a 'B' over 25 images. Image index '12' may be regarded as half way between an 'A' and a 'B'.

The data set could best be represented by a scalar (from the output of a single node) if the pure 'A' and 'B' elicited maximal equal and opposite responses with image index '12' yielding a zero response. Since the images are merged linearly, the code produced by the node output should also change linearly from image index '0' through to '25'.

The response of the trained node to this data set is shown in fig 4.1. We can see that, for learning rates less than 0.0025, the node *performs well as a dimensionality reducer*. The node response is not completely linear however and exhibits a slightly sinusoidal characteristic which does not quite preserve the distance relationships in the original domain. This effect is most marked where the magnitude of the node output is large. We can also see that the learning rate has an effect upon the performance of the node where learning rates in excess of 0.0025 cause the system to converge to non-optimal solutions.

At this point it may be instructive to investigate how some of the system parameters influence the learning rate so that we might choose the most appropriate ones for our particular application.

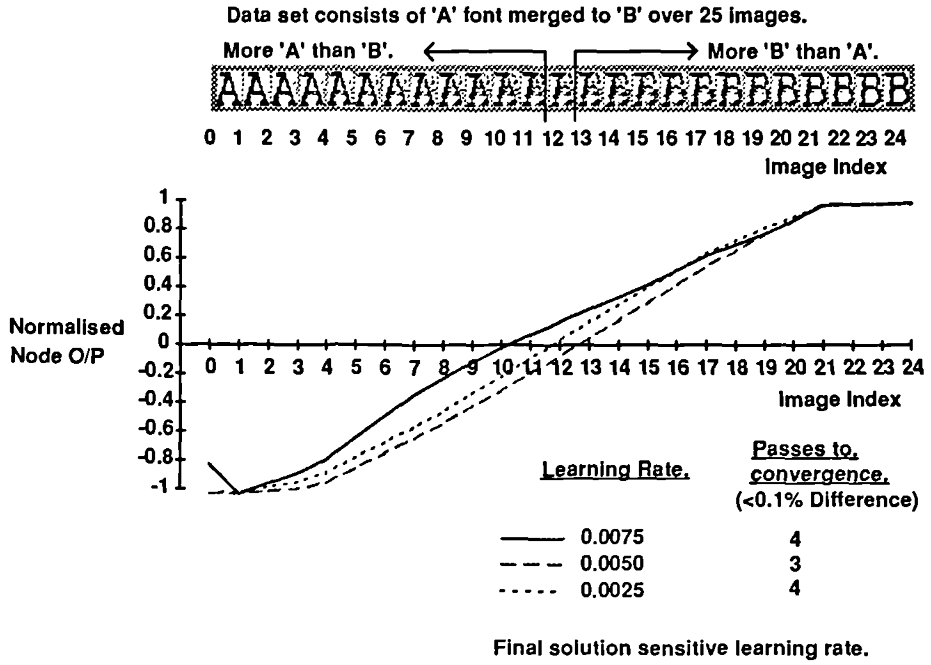


Fig. 4.1 A single constrained Hebbian node as a dimensionality reducer

4.2.2. Learning rate and the dynamic behaviour of the constrained Hebbian node.

Setting the learning rate in neural network systems is seldom an arbitrary affair. In the derivation of the update rule for constrained Hebbian learning, given in equation (18, we can see that several assumptions have been made, notably in equations (14 and (16 , which render it valid only as long as $\alpha \rightarrow 0$. As α becomes large, higher order effects will begin to feature in the system dynamics which could upset the performance of the network. Of course, this could be circumvented rather easily by setting α very small but this incurs commensurately long training times which are not always practical. Some compromise must be met here.

4.2.3. A practical test to assess the performance of an image retrieval mechanism.

The following test provides an objective measure of the performance of an image retrieval mechanism. The data set, shown below in fig. 4.2, consists of 10 classes of machine printed 16 by 24 bit binary characters 'A' to 'K' (not including 'I') with 10 characters per class.

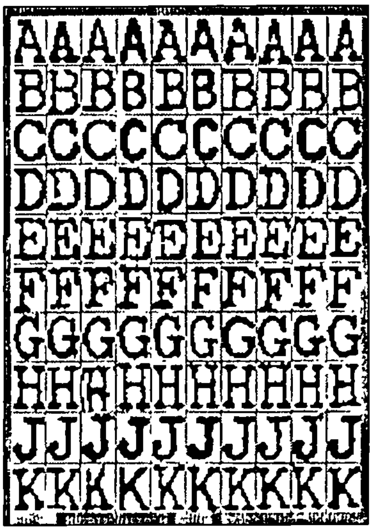


Fig. 4.2 Data set for retrieval experiments

The complete data set is presented to the network during training until convergence, deemed to occur when the respective outputs for all given patterns changes by less than 0.1% for subsequent presentations of the data set. The code is taken to be the output of the node after convergence.

Images are matched against a target image, similar but not identical to those already contained within the database, by retrieving those images whose codes are closest in the Euclidean sense to the target code. The number of images extracted from the database for each target corresponds to the number of examples in each class, in this case 10. The object of the retrieval mechanism is to retrieve all images from the same class as the target and no other class. An error is defined as a class mismatch between the target and any one of the 10 retrieved images. The test is repeated for 10 examples each of 10 target classes (corresponding to the classes contained within the database). The total number of errors is defined as the number of mismatches for the complete data set. Thus, the maximum total error for this data set is 100 x 10 = 1000.

4.2.4. How does order of presentation affect training?

The learning rate may be viewed as a method of diffusing the context of a particular pattern across the data set. This prevents the network from learning localised anomalies arising from a particular ordering of the data set. As an example, consider training the node with two classes of 'A' and 'B' characters in the two orders shown below:

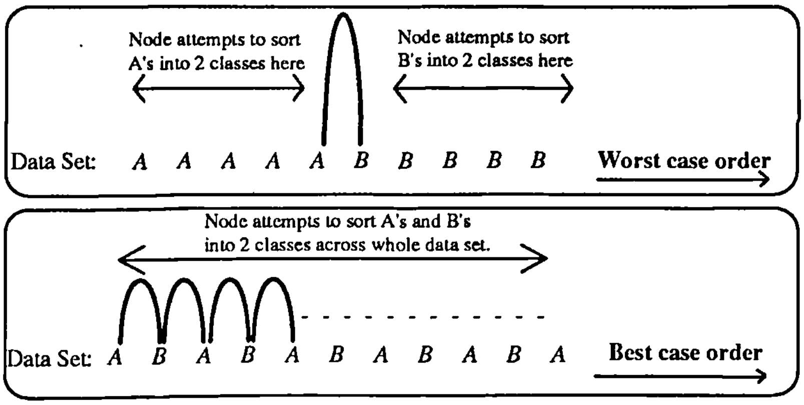

Node attempts to sort A's into 2 classes here
Node attempts to sort B's into 2 classes here
Data Set: A A A A A B B B B B
Worst case order
Node attempts to sort A's and B's into 2 classes across whole data set.
Data Set: A B A B A B A B A B A
Best case order

Fig. 4.3. Effect of training set order on learning performance.

Figs. 4.4 and 4.5 show how the performance of the system is effected by the ordering of the training data. An interleaved and non-interleaved class presentation may be regarded as $(A_1,B_1,C_1,..A_2,B_2,C_2,..)$ and $(A_1,A_2,A_3,...B_1,B_2,B_3,...)$ respectively.

Bearing in mind that the objective of constrained Hebbian learning is to maximise the standard deviation of the node, fig. 4.4 shows that low learning rates will prevent the system from converging to non-optimal localised solutions arising from a particular ordering of the data set. This set of graphs also indicates that learning is most effective when consecutive training patterns are maximally dissimilar. Training in a practical system should reflect this.

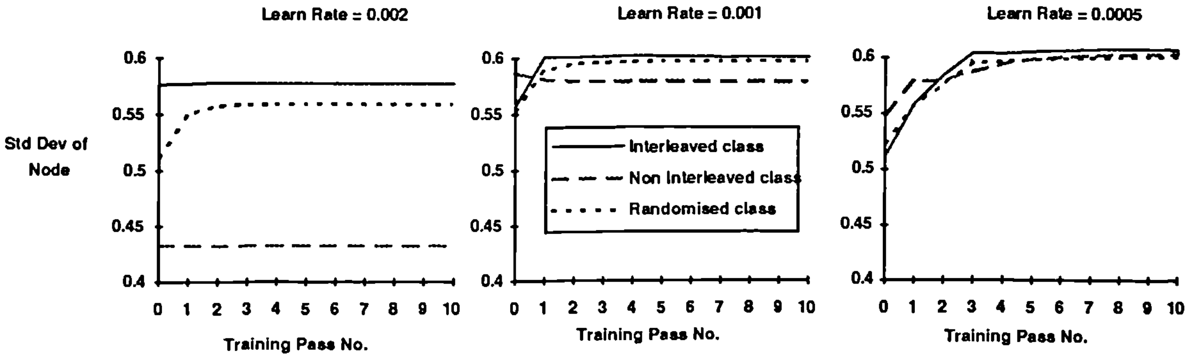


Fig. 4.4 Effect of training data order on convergence and variance of the Node.

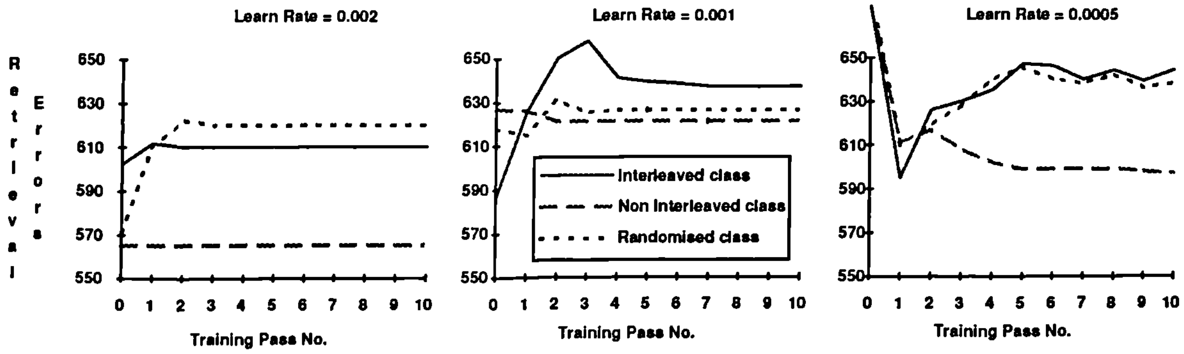


Fig. 4.5 Effect of training data order on performance of a single node coding system.

However, fig. 4.5, which shows the actual performance of a practical retrieval mechanism for this test, indicates that the standard deviation of the node does not provide such a concrete measure of the performance for this data set as theory would suggest. This result is contrary to the thrust of PCA which maximises the variance of the node in an attempt to optimise the amount of information captured by the transformation developed during training - frequently referred to as mutual information. It is reasonable to suppose that a transform which maximised the mutual information *per se* would yield the most effective coding scheme for that data set. Reasonable, but not quite correct. Linsker *[56]* shows that maximising the variance will only maximise the mutual information if the data comes from a Gaussian distribution; if this is not the case then PCA produces a sub-optimal information preserving transform. It is apparent from the results presented in fig. 4.5 that our data set does not satisfy this criteria. In fact, the performance of the retrieval mechanism seems inversely proportional to the standard deviation within the limited range of parameters of this test. The implication of this result is that the Standard Deviation does not provide a clear picture of the potential performance of the code for constrained data sets. This makes it difficult to optimise the system parameters for our scheme. As the database becomes more diverse, so that the distribution of images tends to a Gaussian, the standard deviation provides an increasingly accurate indication of retrieval performance.

4.2.5. Dimensionality and learning dynamics.

An investigation of how the learning mechanism responds to the dimensions of the training set might allow us to match the learning rate to the nature of the problem at hand. The objective of this and the next few tests is to attempt to avoid the rather *ad hoc* approach to system parameter selection which has often dogged neural network applications in the past.

The test data consists of 3 sets of images, derived from the data set shown in fig. 4.2, which are identical in every way except that the images for subsequent sets are scaled up (in area) by a factor of 4. In this experiment the learning rate is chosen so that the system converges rapidly, but asymptotically, to a stable solution. The test is repeated for the remaining two data sets, adjusting the learning rate until the convergence profile is identical to the first. Note that the Standard Deviation is obtained from the normalised output of the node to permit some comparison between the data sets. The results are shown below in fig. 4.6.

Data Set Dimension	Learning Rate	No. Passes to Convergence	Standard Deviation	Retrieval Errors.
64 x 96	0.00006	8	0.5966	625
32 x 48	0.000235	6	0.5967	626
16 x 24	0.000975	5	0.5965	626

Fig. 4.6 Effect of pattern size on convergence and performance of code.

These results show that the system scales up very well - its performance is insensitive to the size of the images being encoded. They indicate that the learning rate should be changed according to the size of the images within the database - if the image size is doubled (in area) then the learning rate should be halved and so on.

4.2.6. Data set size and learning dynamics.

The objective of this test is to observe how the encoding scheme behaves as the data set increases. To allow a fair comparison between data sets, we must attempt to de-couple data set size from diversity by ensuring that, as it grows, the diversity remains fixed. This is achieved by fixing the number of classes of characters contained within the database, changing only the number of characters per class. In this test the relative errors are defined as $\frac{No.\ of\ Errors}{Max.\ possible\ Errors}$. The results are shown below in fig. 4.7.

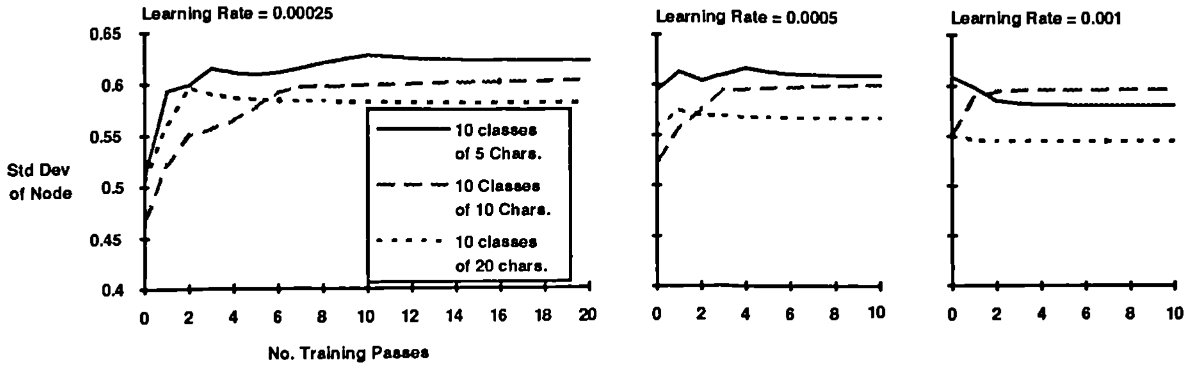


Fig. 4.7 Effect of data set size on convergence and variance of the node.

These results show that large database systems will converge more rapidly than smaller ones but are prone to settle into non-optimal solutions (if we regard an optimal solution as one which yields the highest Standard Deviation). This is because, for large data sets, the learning must be 'spread' over a larger area to ensure that the node extracts global rather than local characteristics from the ensemble. This follows from the discussion detailed in 4.2.4. Thus, to maintain the same performance the learning rate must be lowered as the database expands.

The test outlined in 4.2.5., which matched the learning rates required to achieve equivalent performance, is repeated for a range of data sets. Results, shown in below fig. 4.8., indicate that the learning rate should be made inversely proportional to the size of the database; that is, if we double the size of the database then the learning rate should be halved.

Data Set Size	Learning Rate	No. Passes to Convergence	Standard Deviation	Relative Retrieval Errors.
10 Classes of 5	0.00004	102	0.6375	0.5760
10 Classes of 10	0.00002	97	0.6019	0.6350
10 Classes of 20	0.00001	91	0.5963	0.6595

Fig. 4.8 Effect of data set size on performance of a single node coding system.

4.3. A Multi-Node Coding Scheme using Constrained Hebbian Learning.

It is rather optimistic to expect the vast amount of information contained within an image database to be represented adequately by a scalar valued code. We must expand our scheme to a vector code, achieved by increasing the number of nodes within the system so that the output from each node forms an element of that vector.

Fig. 4.9 indicates how the system converges for a range of learning rates. We can see that the number of passes required for convergence is directly proportional to the learning rate for multi-node systems - this is especially evident in node 9.

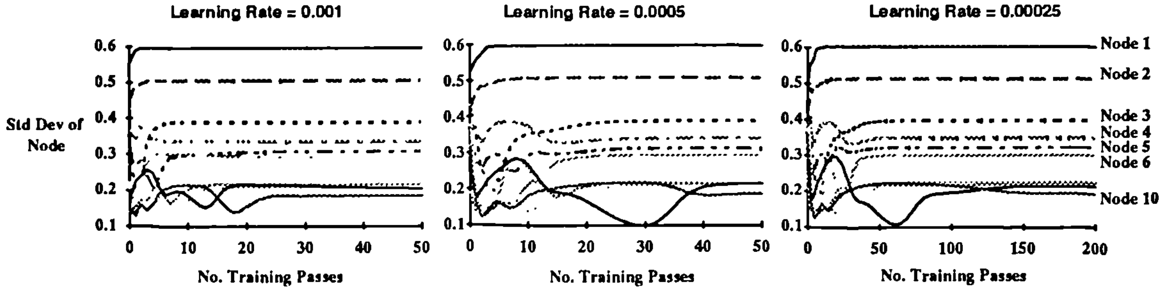
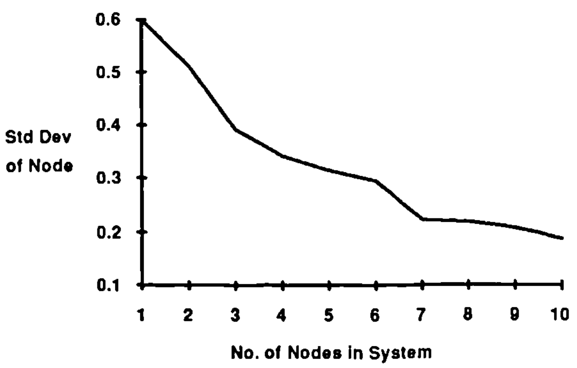


Fig. 4.9. Convergence for multi-node constrained Hebbian learning.

For a data set with a Gaussian distribution, the standard deviation of the node gives an indication of the amount of information that the node is contributing to the description of that data set.

Successive nodes provide progressively less information until the addition of additional nodes hardly furnishes us with any extra information at all. For this particular data set 6 nodes seems to be optimum. The relationship between the standard deviation, code performance the number of nodes is shown in figs. 4.10 and 4.11 respectively. The latter shows that extending the system beyond 6 nodes reaps meagre rewards.



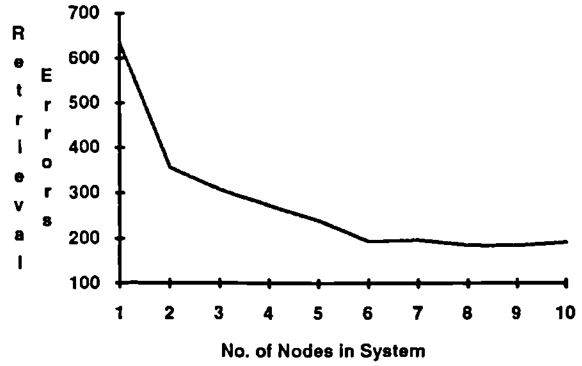


Fig. 4.10. Variation of standard deviation of O/P with node index for a constrained Hebbian system.

Fig. 4.11 Relationship between number of nodes and code performance.

The degree of similarity between the breaks in the response profiles in figs. 4.10 and 4.11 suggests that the standard deviation could be used to select the optimum number of nodes for any given data set.

A graphical representation of the weights after convergence, *i.e.* the Principal Components of the data set, are shown in fig 4.12. The darker areas indicate regions of high negative importance (*i.e.* large negative weights) and the lighter ones regions of high positive importance (*i.e.* large positive weights). The grey areas do not contribute significantly to the node output. These 'masks' exhibit recognisable characteristics prevalent within the database which become less distinct for successive nodes. Nodes 7 to 10 are, ostensibly, remarkably similar (except that node 10 is the approximate inverse of node 7) which consolidates the findings shown in figs. 4.10 and 4.11.

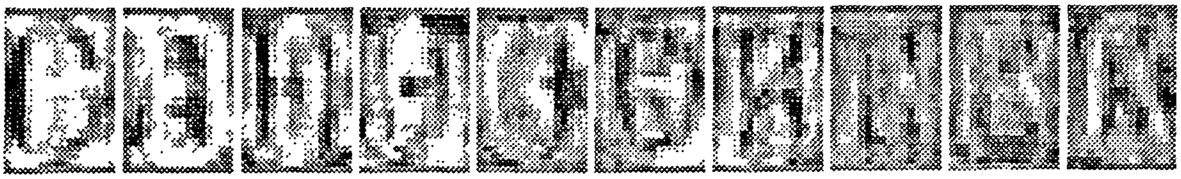


Fig. 4.12. First 10 Principal Components (ordered left to right) for data shown in fig. 4.2

4.4. Database Contents and System Performance.

4.4.1. Matching code length to diversity of database.

The results presented in 4.3 suggest that the number of nodes, corresponding to the size of the code, can be optimised with respect to the nature of images within the database. The objective of the following tests is to show how the code length can be matched against the diversity of images within the database. In the first test, the diversity of the data base is increased by adding a further 10 examples of a new class of characters into the database and re-training the system from scratch. The results are presented below in 4.13.

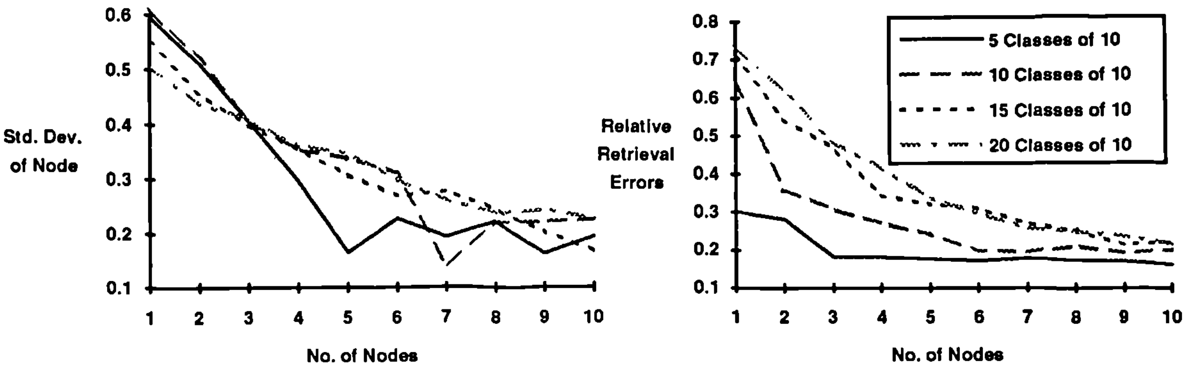


Fig. 4.13. Variation of performance with code length (*i.e.* number of nodes) for progressively diverse databases.

It is instructive to see how the performance of the system is affected if we add images similar to those already contained within the database. To this end, the earlier test was repeated but, instead of adding images from new classes, we add further examples of images already contained within the database. The results are shown below in 4.14.

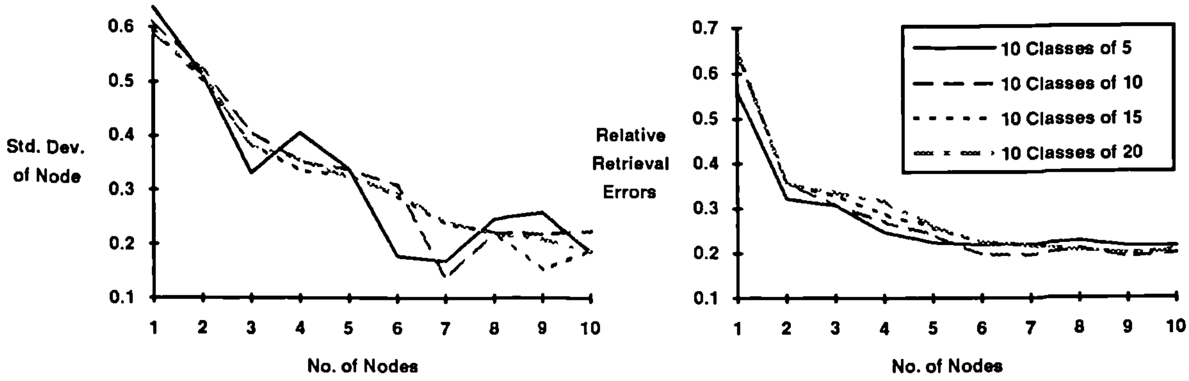


Fig. 4.14. Variation of performance with code length (*i.e.* number of nodes) for progressively larger databases of constant diversity.

Not surprisingly, we can see from the results that the more diverse the data set the larger the code required to achieve the same level of performance. Fig. 4.14 shows that performance of the code is not markedly effected by adding images similar to those already within the database. Of course, the size and diversity of a database are inexorably linked and it is likely that adding an image will have some effect upon its diversity.

Fig. 4.13 indicates that, as the database becomes increasingly diverse, the improvement in performance afforded by additional nodes becomes increasingly marginal. This suggests that as the database becomes sufficiently rich in image types, the nodes will begin to learn a set of generalised feature primitives which are capable of describing global image archetypes. Such primitives are common to a broad range of images.

This is the thrust of the Hadamard Transform and Discrete Cosine Transform (DCT) *[59]* which use a fixed set of local masks which are weighted and superimposed to reconstruct an image. The potential for coding here lies in the fact that only the weighting coefficients for each mask need be recorded of which relatively few are required for a faithful rendering of the original. We can see that this technique has the same 'feel' as our own coding mechanism. However, the primitives in our own coding scheme adapt to the contents of the database whereas those for the Hadamard and DCT remain fixed. In an attempt to represent the images in a lower dimensional bound, PCA attempts to extract both spatial and ensemble correlations from the data set. It utilises second order statistics. By contrast, DCT and the Hadamard Transform, for example, do not take ensemble characteristics into account and rely only on first order statistics. Thus, our coding mechanism is likely to produce higher performance codes than either of these methods, especially if the data base contains sets of highly correlated image types.

It is worth pointing out here that any coding strategy only becomes economic if there are 'recurring themes' that occur within the data. These represent redundancy and redundancy gives us scope for dimensionality reduction which, in turn, provides coding potential. They manifest as ensemble themes or as spatial artefacts. Such themes might be characterised by the class of data within the data base (such as letters, faces or finger-prints, for example) or, at a more fundamental level, by virtue of the fact that meaningful real-world images contain common spatial primitives and such as continuous line segments or areas of constant luminance, for example.

This latter case was investigated by Linsker *[56]* using a neural network and learning law practically identical to the one proposed for our system. He showed that when trained with a broad range of image types, the weights converge to feature primitives which characterise the general underlying characteristics common to all image types. This work is exciting in that these artefacts bear a remarkable resemblance to the so-called 'retinal fields' found to exist within mammalian visual systems. These fields, located within the primary visual cortex, are used to encode images for subsequent processing stages 'higher up' in the cortex. This seems to validate the approach of our own coding methodology.

4.4.2. The need for re-training.

The proposed coding strategy represents each image in terms of its relationship to a feature, or set of features, which characterise the spread of images within a database. As images are added to or removed from the database it is likely that the distribution of patterns will change. This, in turn, will alter the nature of the features which permit optimal descriptions of the images therein. If the contents change significantly then the nodes will have to be re-trained to re-define the coding transform. The question we wish to address at this juncture is: 'As the database changes, both in size and diversity, when does it become necessary to re-train the system ?'.

The following example is illuminating: If we had a database of 1000 faces, for instance, it is quite likely that practically all of the salient features which characterise a face would be adequately represented by the existing data set. Consequently, adding a few more faces to the database is unlikely to change the coding transform significantly and the new images could be coded without re-training. If we wished to add a *different class of* images to the database, however, then the need for re-training is more pressing. For instance, if we added several images of horses to the facial database then it is unlikely that a horse could be adequately described in terms of features extracted from human faces (although a few exceptions do spring to mind!).

4.4.3. Re-training for a database containing limited image types.

In this test a 6 node system node is trained with the data set shown in fig. 4.2 until convergence (learn rate = 0.00004; where convergence is deemed to occur when respective node responses differ by less than 0.1% on successive passes). The contents of the database are then changed but its diversity fixed by keeping the number of classes of characters within the database constant. Note that the node is not re-trained; the new images are coded using the existing weights.

The relative retrieval errors and the standard deviation of the first node, which appears to give the best indication as to the potential performance of the code, are shown below in fig. 4.15. The training times required for re-training using the existing weights and 'starting from scratch' are also shown.

Data Set Size	Relative Errors (no re-training)	Relative Errors (with re-training)	Std Devn of node 1 (no re-training)	Std Devn of node 1 (with re-training)	Passes to converge (no weight initialisation)	Passes to converge (with weight initialisation)
10 Classes of 5	0.2120	0.2080	0.5985	0.5839	33	94
10 Classes of 10	0.1940	-	0.5685	-	-	46
10 Classes of 15	0.2231	0.2249	0.5687	0.5899	16	33
10 Classes of 20	0.2278	0.2278	0.5675	0.5744	14	35

Fig. 4.15 Effect of re-training for expanding database of constant diversity.

This shows that the relative performance of the code remains remarkably constant as the database expands, provided that the images that are added are similar to those images trained into the node. Indeed, if the new images are similar to the existing ones then the improvement in performance afforded by re-training is quite marginal. This is rather convenient as it does away with the need for 'on-line' training which is likely to be quite a time consuming process. A practical expedient would be to re-train the system during a 'down-time' at regular intervals.

For the increase in performance that it produces, the amount of re-training required under these conditions is comparatively large . This is because the learning process is motivated by the difference between the actual weights and the weights representing the optimal feature. If the new images are very similar to the existing ones then the differences which motivate the learning are comparatively small and a large fraction of the total training time is spent near the asymptotes of the learning curve.

4.4.4. Re-training for a database containing unlimited image types.

The experiment outlined above in 4.4.3. is repeated here but instead of adding images from classes already within the database, we add images from completely new classes in blocks of 10 per class (in this case we gradually add characters 'L' to 'V' not including 'O' in blocks of 10). The results are shown below in fig. 4.16.

Data Set Size	Relative Errors (no re-training)	Relative Errors (with re-training)	Std Devn of node 1 (no re-training)	Std Devn of node 1 (with re-training)	Passes to converge (no weight initialisation)	Passes to converge (with weight initialisation)
5 Classes of 10	0.1840	0.1640	0.5820	0.5868	46	49
10 Classes of 10	0.1940	-	0.5685	-	-	46
15 Classes of 10	0.3027	0.2767	0.5276	0.5321	53	69
20 Classes of 10	0.3135	0.2980	0.4916	0.4419	27	87

Fig. 4.16 Effect of re-training for expanding database of increasing diversity.

These results show, as we might expect, that as the database becomes more diverse, the efficacy of the retrieval mechanism begins to drop off. Because the additional images are from different classes to those already contained within the database, the existing weights are likely to produce sub-optimal codes. The results bear this out.

Under these conditions retraining will improve the performance of the retrieval mechanism more than the case mentioned in 4.4.3 but the performance without re-training is still quite good. This suggests that the node has learn a set of image primitives which can be used to describe a broad class of image types, even those not seen during training. The greater the diversity of the database, the more likely it is that the node will begin to learn such primitives. Under these conditions the effects of re-training become increasingly marginal. As the database becomes increasingly diverse the transform developed by the nodes will tend to the Hadamard Transformation *[59]*.

Note that the standard deviation does not provide a 'hard and fast' indication of the potential performance of the code. The implication here is that, as we add images to the database we have no clear notion of the degree of improvement that re-training will bring. However, a rather crude approach might be to re-train the node once the standard deviation dropped by a certain percentage.

4.5. Non-Linearities - A Practical Assessment of Their Worth.

4.5.1. Non-linearities and the prospect for improved performance.

It is generally true to say that a non-linear transform will elicit more 'interesting' and 'useful' features from a data set than a linear one. This is particularly true of classifier systems where we need to isolate those features which provide an 'all-or-nothing' type response as a class membership function.

However, the neural network transformation required by our coding scheme is not that of a classifier system; rather we require our nodes to exhibit a graded response which preserves the relative distance relationships in the original domain (this point is discussed in 4.2.1 and in 5.3.1 in the context of a logical neural network). It is common practice to insert a sigmoidal transfer function at the output of sum-of-weights to impart some non-linearity into the response. These non-linearities allow arbitrarily complex decision surfaces to be formed with multi-layer architectures (by contrast, a linear transformation will only ever produce a planar decision surface, irrespective of the number of layers). A range of non-linearities based on the hyperbolic tangent function is shown below in fig. 4.17. the actual transfer function is given by:

$$f_{sigmoid}(\ ) = \frac{\text{maxoutput}}{\tanh(\text{sigmoid\_coeff})} \times \tanh\left(\frac{\text{output} \times \text{sigmoid\_coef}}{\text{maxoutput}}\right) \qquad \text{(23}$$

This constrains the output of the node to lie within the limits of its linear counterpart.

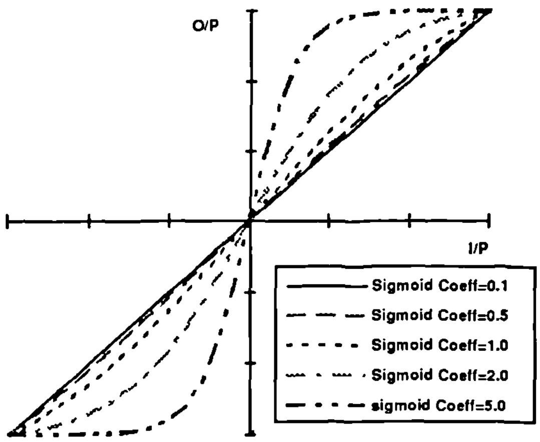


fig. 4.17. Non-linearities used for experiments in 4.5.1.

To observe the effect of this non-linearity we can repeat the experiment outlined in 4.2.1. with equation *(23* applied to the output of the node (both in the coding and training phase). The results are shown below in fig. 4.18.

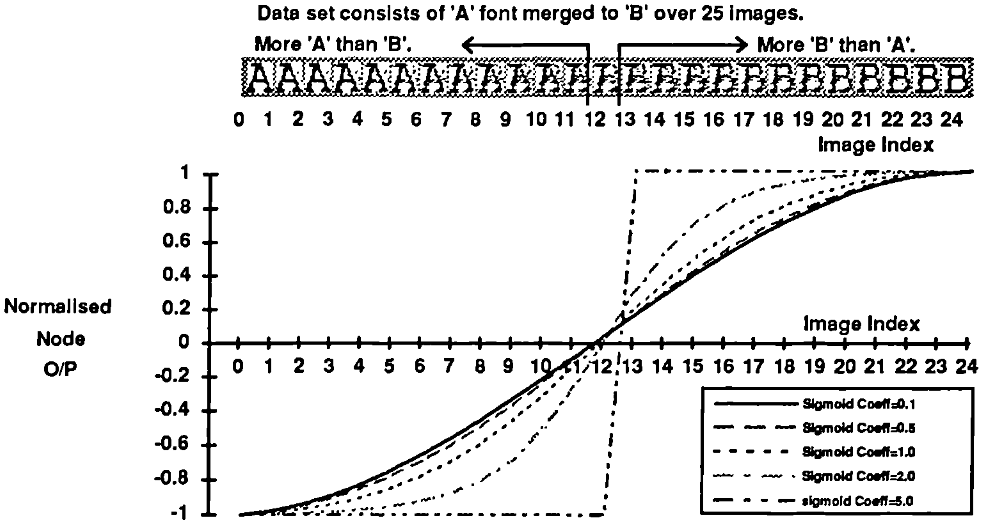


fig. 4.18. Increasing non-linearity changes node from a dimensionality reducer to a classifier.

These show that as the non-linearity is increased (by changing the coefficient from 0.1, a near-linear response to 5.0, a near step response) the node begins to function more like a classifier, dividing the data set cleanly in half.

Fig. 4.19 indicates that the learning in non-linear nodes is highly sensitive to learning rate and increases the likelihood that the system will settle into a false minima. It is rather interesting to note that the system takes much longer to settle into a sub-optimal solution even thought the learning rate is higher.

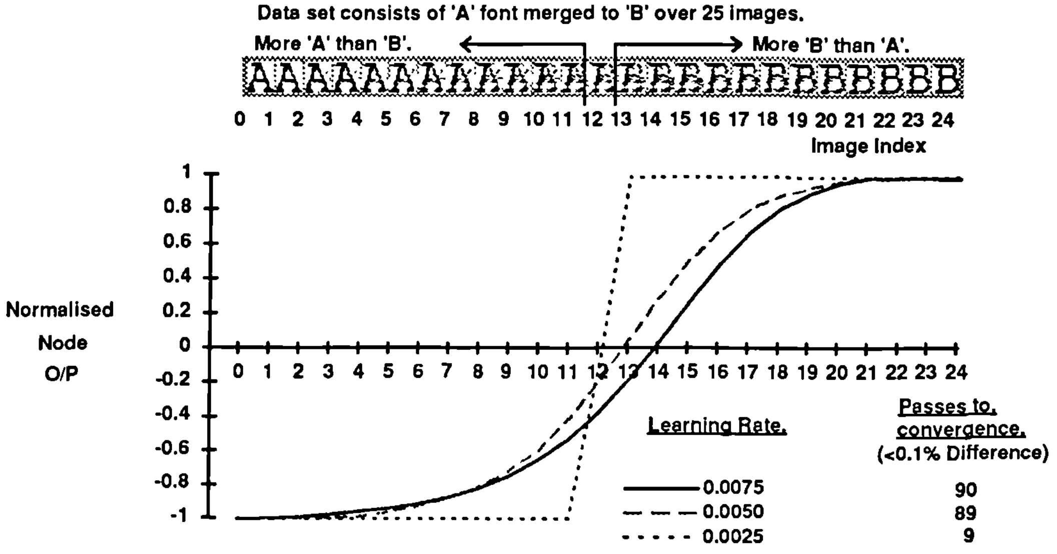


fig. 4.19. Increasing the non-linearity makes the system sensitive to the learning rate. (shown here with sigmoid = 2.0).

So, how does this non-linearity effect the performance of our code? The experiment described in 4.3 is repeated for a 6 node system (deemed to be optimal for this particular data set) with a range of non-linearities. The results are shown in fig. 4.20 - note that for sigmoids > 2.0 the retrieval errors were in excess of 500 throughout training during which the system failed to converge.

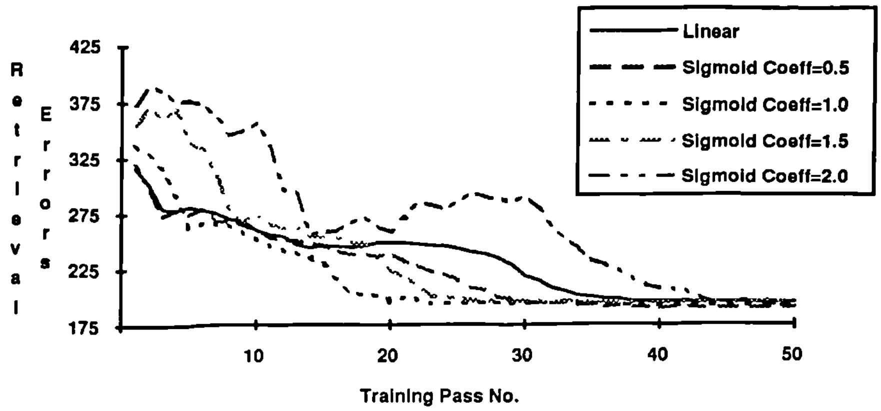


fig. 4.20. Effect of non-linearity on convergence and performance of code (shown here with learn rate = 0.00025).

It is interesting to note that a small non-linearity does not effect the performance of the code to any great extent but does assist in the training phase as convergence tends to be speeded up a little. This is because the non-linearity has a tendency to 'push' the output of the node to the limits and in so doing 'pushes' it that little bit faster toward a solution. The fact that this non-linearity has only a marginal effect upon the performance of the code surprises this author a little.

Following the arguments presented by Bourland and Kamp *[58]* in chapter 3, we suggested that non-linearities were not particularly advantageous for the coding scheme. Whilst our own results corroborate this reasoning in that they show that non-linear transforms offer no improvement in performance, they do indicate that a degree of non-linearity is useful in that it improves the convergence of the system. This seems to counter the argument forwarded by Bourland and Kamp which implied that training times are likely to increase for non-linear systems.

4.6. An Intelligent Video Editing System using the Neural Network Coding Scheme

The coding scheme investigated in this chapter can be used as the basis of an intelligent video editing system. Video editors are frequently required to access sections of a video sequence that contain a particular scene - this is, of course, just a re-interpretation of a retrieval by image content system. It is illustrative here to show how our coding scheme performs for a typical sequence of 'newsreel' footage. The video editor selects a particular scene and requests that the system retrieve those scenes that bear the closest resemblance to the target.

4.6.1. Methodology

In the following example, a single frame of broadcast news footage is grabbed and stored each second. The image is converted to 8-bit grey level, shrunk from 512 x 512 to 128 x 128 and stored on disc; in a practical system the images need not be stored since the network could be trained on the 'raw' video. This was repeated for 400 images - the complete database is shown in Appendix A (the apparent poor resolution of these images is a product of the printer used here. They are best viewed at a distance).

Ten nodes were trained on all 400 images with the learning rate set to 0.0001. The convergence of each node is shown below in figs 4.21 and 4.22. The latter indicates that the training times increase approximately linearly with the number of nodes.

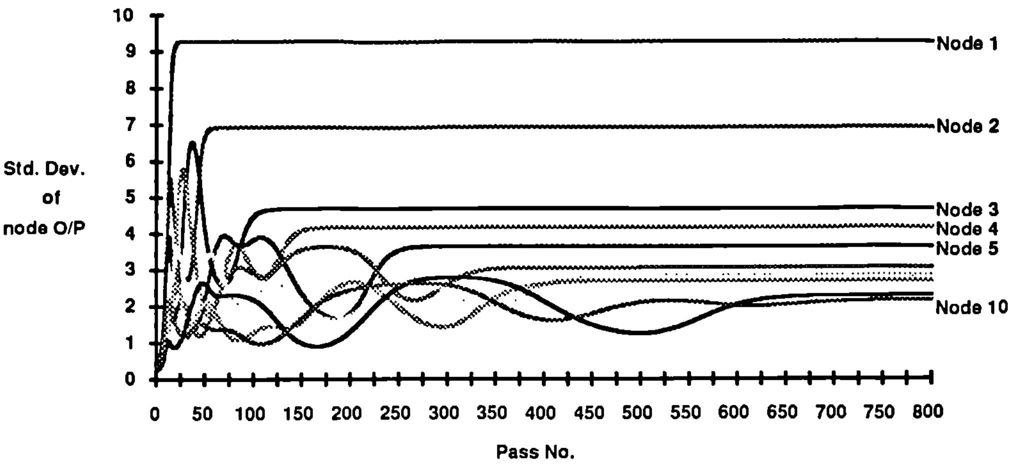


Fig. 4.21. Standard deviation of nodes during training.

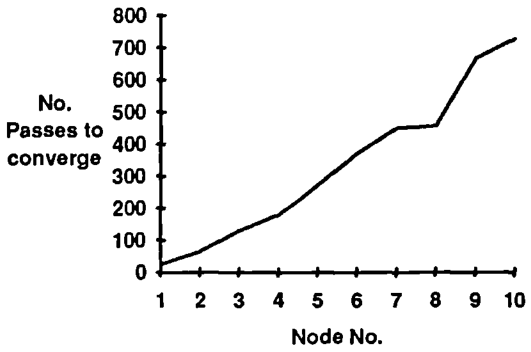


Fig. 4.22. Convergence times for nodes during training.

The output of each node was quantised to 8 bits so that the code for each image was 80 bits.

4.6.2. Results

A set of target scenes were selected from the database and 25 frames with the closest match retrieved. The targets were chosen to present something of a challenge to the retrieval mechanism (for example, the head and shoulders of the news-reader appear frequently throughout the footage with only the relatively small inset in the top right hand corner to distinguish it from other frame sequences). The results are shown below in figs. 4.23 to 4.27; in each case the target image and the image with the closest score is shown in the top left of the diagram as image index 0. The retrieved images are ordered left-to-right and top-to-bottom.

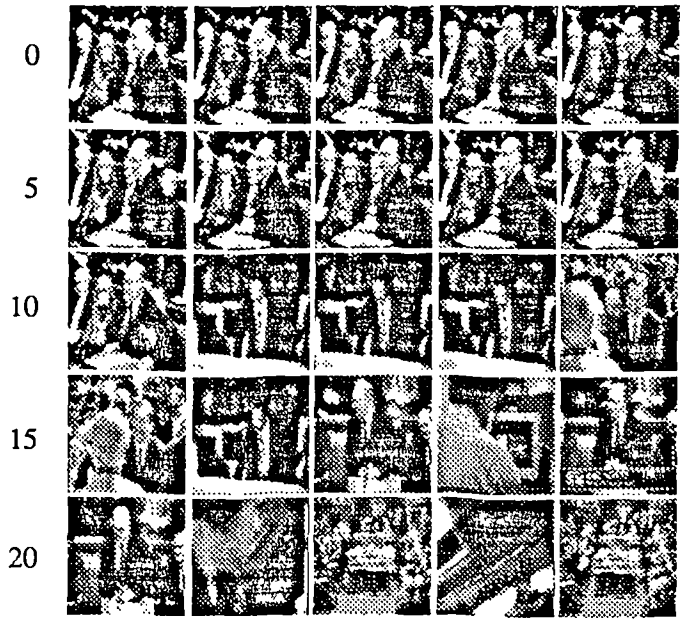


Fig. 4.23. Target image index:60. Retrieved images 60,61,63,62,65,67,66,64,69,68,70 ,156, 158,157,164,163,159,34,360,33,35,11,47,13,46.

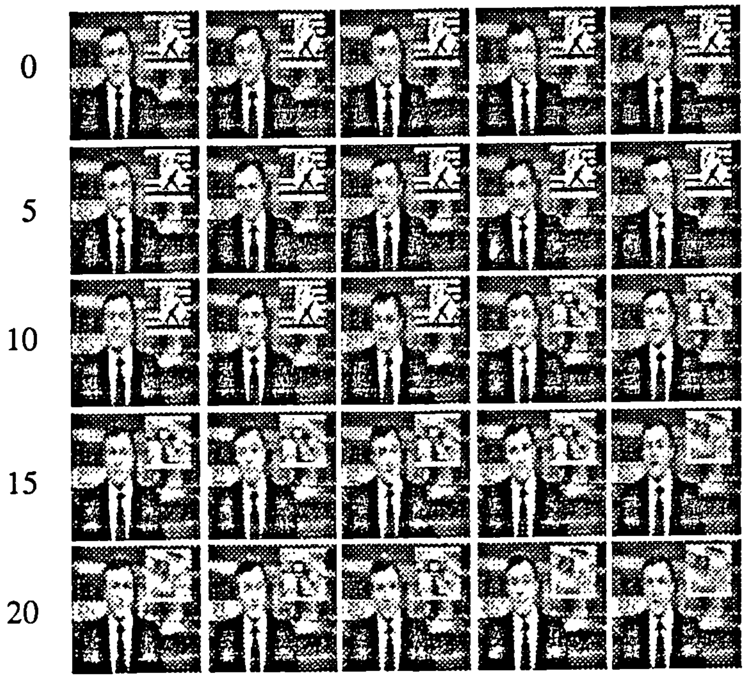


Fig. 4.24. Target image index: 89. Retrieved images: 89,100,98,92,99,96,94,91,95,97, 93,90,101,323,327,326,324,325,328,220,217,330,329,221,224.

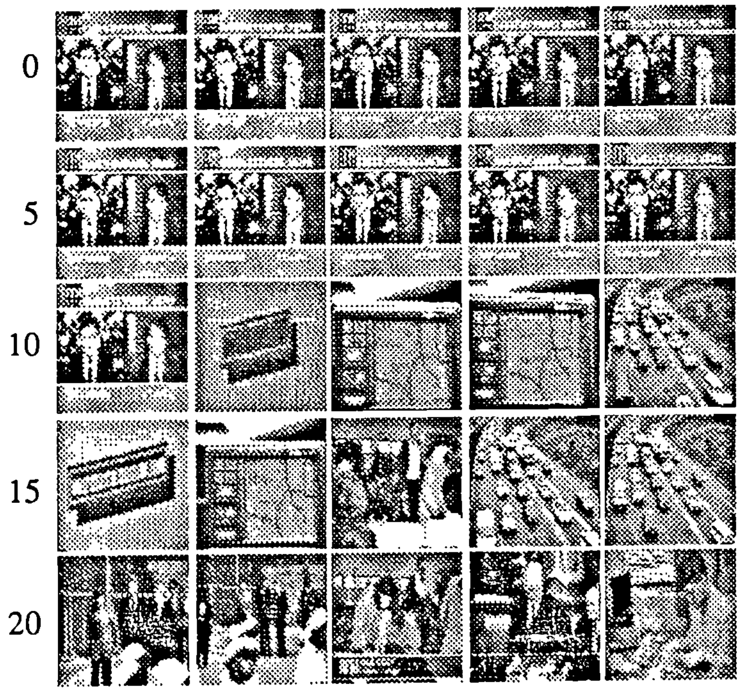


Fig. 4.25. Target image index: 166. Retrieved images: 166,169,187,167,188,210,211, 168 ,170,190,189,261,338,337,350,262,339,107,352,351,144,143,106,,40,254.

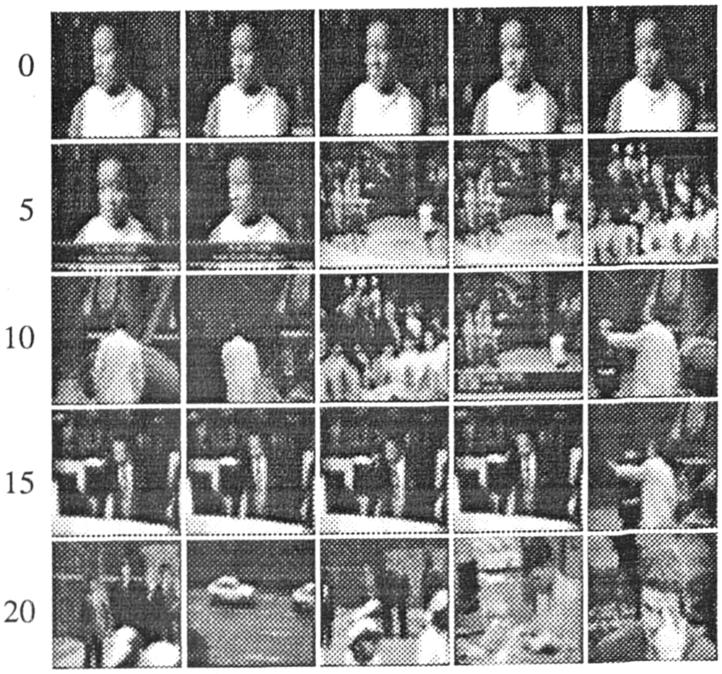


Fig. 4.26. Target image index: 299. Retrieved images: 299,304,305,302,303,300,301 ,377,378,322,114,117,321,379,115,157,156,159,158,116,144,357,143,254,391.

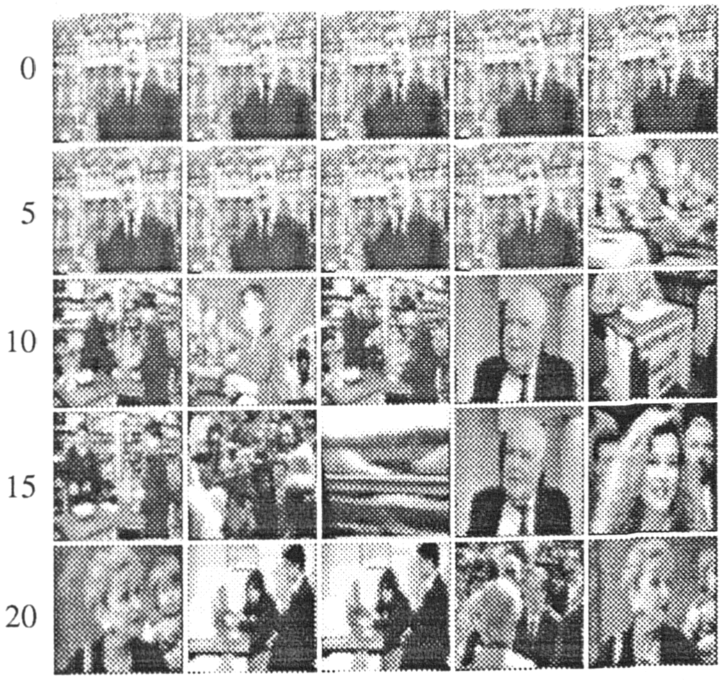


Fig. 4.27. Target image index: 306. Retrieved images: 306,308,312,314,307,311,310, 309,313,231,246,230,247,140,104,242,160,341,141,384,113,264,265,163,111.

4.6.3. The effect of the number of nodes on the performance of the editing system.

Reducing the number of nodes is beneficial because it speed up both the search and the training times. The next test gives some indication of the relationship between the number of nodes and the performance of the editing system.

Here image index 89 is used as the target (see fig 4.24) and the previous test repeated whilst increasing the nodes from 1 to 10. Image index 89 is particularly apt for this test since the only artefact which differentiates it from other shots of the news readers' head is the small caption insert in the top right hand side of the frame. The sequence beginning with index 89 continues for 13 successive frames, whereupon a different sequence begins. The object of the retrieval mechanism is to recover all 13 of these frames and no others.

The results are shown below in fig. 4.28 where the confidence of the retrieved images runs left-to-right. Note that image index 89 is always the first to be retrieved, regardless of the number of nodes and is shown as the first image of the retrieval sequence. Both the 8 and 10 node systems recovered all 13 of the target frames. It is evident that an 8 node system is perfectly adequate for this particular data set.

4.7. Discussion

Neural networks are frequently regarded by the uninitiated as a kind of computational panacea. It is all too easy to get the impression that one merely has to 'throw' data at an off-the-shelf architecture, leave the neural network to 'get on with it' and come back a short while later to find the problem solved. In reality though, this is simply not the case and a neural network practitioner must typically ponder long and hard over a host of frequently baffling network parameters before even beginning to tackle the problem at hand. Such parameters might include network topology, nodal transformation function, learning law and its associated parameters such as the learning rate, which we have already touched upon. This is not an easy task and the selection process is often closer to alchemy than it is to science. Even then, there is no way of knowing that the solution found by the neural network is an optimal one. This is tantamount to saying that neural networks can be used to solve a problem provided that you already know what the solution is! This is not an issue for supervised learning systems where we have some idea of the classification boundaries but is a problem for unsupervised learning systems where there is no *a priori*

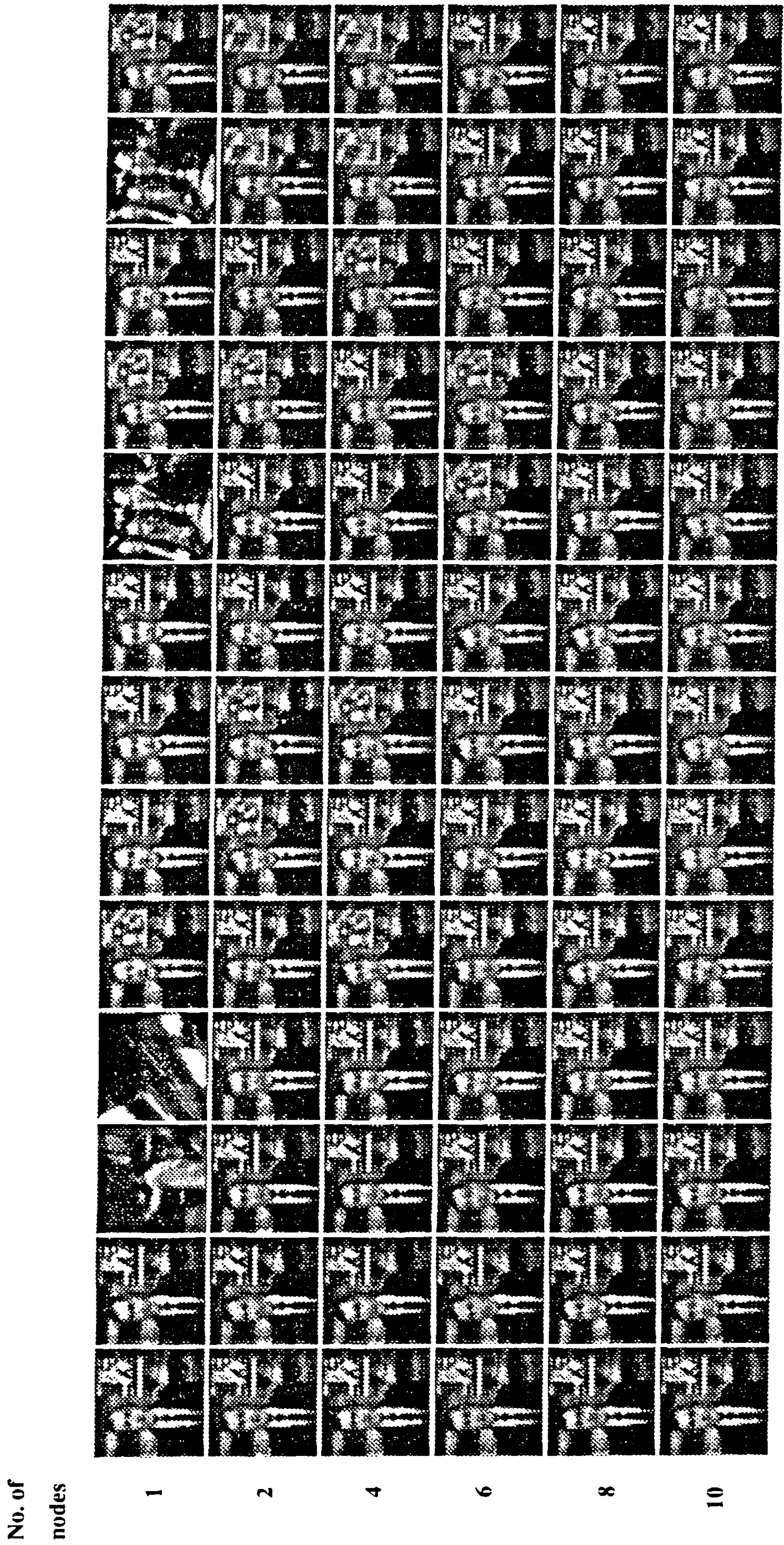


Fig. 4.28. Increasing the number of nodes improves the performance of the retrieval mechanism.

knowledge of notional data organisation. This state of affairs is unacceptable for a practical system where the neural network is to be trained by an operator with only modest expertise. Throughout this chapter we have attempted to address this problem with a view to automating the selection of the network parameters for the neural network and constrained Hebbian learning law introduced in chapter 3. To this end we have investigated how the parameters need to be adjusted as the database changes. The number of nodes and the learning rate are two particularly important parameters in our network.

The learning rate is critical and the surest way to guarantee good performance is to set it very low. Unfortunately, this means that training times are very long and some compromise must be met here. Work outlined in this chapter indicates how the learning rate may be set and changed to suit the nature of the data base.

The size of the learning rate determines the extent localisation of the learning process within the training ensemble. If the rate is set high then the system has a propensity to learn local features dependent upon a particular ordering of the data during training. Such a system is likely to converge to sub-optimal solutions. A small learning rate will 'spread' the context of the learning across the data set. We show that the learning should be made inversely proportional to the size of the data set.

The size of the images does not effect the performance of the code - we show that the learning rate should be made directly proportional to the size of the images within the database. Of course, this is rather a simplistic notion as larger images tend to contain more information than smaller ones and this would undoubtedly degrade the performance of the system. Note that this system requires that the size of the images be normalised so that they all have the same dimensions.

To a limited extent, the standard deviation of the node activity across the data set can provide a metric to help us assess the efficacy of the retrieval mechanism; though the relationship is a loose one and does not always hold true. Indeed, we point out that situations can arise whereby the system can settle into a solution which exhibits a higher standard deviation but with a lower performance. Since low learning rates will always maximise the standard deviation we can conclude that an optimal PCA may not necessarily provide the best retrieval performance for a given data set. The performance will only be optimised if the data has a Gaussian distribution in the pattern space. Fortunately, most images exhibit Gaussian like statistics and the performance may be reckoned to be near optimal for a large and sufficiently diverse database.

Successive nodes learn features with decreasing significance. For any given data set there comes a point where the inclusion of additional nodes provides little or no additional information. Thus, it is possible to optimise the code length with respect to the contents of the data base. For a database of moderately diverse image types (so that the distribution tends to that of a Gaussian) the standard deviation can be used to select the optimal number of nodes which occurs where successive nodes exhibit standard deviations of the same order of magnitude.

As the diversity of the database increases, the nodes begin to learn generalised image primitives such as continuous line segments, for example. The upshot of this is that the need for re-training is less pressing as these primitives are pertinent to a very broad range of images.

One of the great strengths of our coding scheme lies in its ability to adapt to the changing nature of the database. As new images are added to the database there will come a time when the nodes need to be re-trained although the degradation in performance is gradual and, where the database contains a broad range of image types, almost imperceptible. This begs the following question: 'When should we re-train the system?' Our experimental results indicate that the rate of decrease of the standard deviation of the first node can give a very rough idea of the resultant drop in performance, although the relationship is, again, a loose one. Sometimes the standard deviation can drop by a large amount and the improvement in performance that re-training brings be comparatively small. This is dependent upon the changing distribution of images in the pattern space.

We have looked at the effect of the nodal transformation and show that a simple linear node will capture most information within the data set. These results are backed up theoretically by Bourland and Kamp *[58]* who conclude that non-linear transforms will not outperform linear ones for this type of application. However, whilst our own results show this to be the case they do indicate that a small non-linearity can speed up the training times significantly.

Summing up then, we can see that a simple single layer linear (or near linear) neural network can be used as the basis of our coding mechanism. The transform provided is equivalent to PCA but is less expensive both in terms of memory and computation time. The price that we pay for this is that a learning rate needs to be set which is not an easy task and it is better to err on the side of a small value.

Finally, we show how this coding scheme can form the basis of an 'intelligent' video editing system which facilitates very rapid access to frames with similar scene content.

In chapter 6 we introduce a completely novel learning law which is also directly equivalent to PCA yet requires no learning rate to be set. In this respect it is a more attractive prospect than the system described here.

Chapter 5

Logical Neural Networks

5.1. Introduction.

The objective of this chapter is to show how a single layer Logical Neural Network (LNN) can be used as the basis for a very simple coding scheme and to contrast its performance with the sum-of-weights paradigm discussed in chapters 3 and 4.

We introduce the WISARD architecture and investigate, both through mathematical analysis and experimental results, how the network parameters effect the performance of the code. Such parameters include tuple size and mapping strategy, training methodology and the number of nodes used. We show how a very basic unsupervised LNN coding mechanism can be used to retrieve images from a database of machine printed characters.

5.2. Why use Logical Neurons ?

Architectures for sum-of-weights type neural models abound and most current research in the field of neural networks is centred around implementing learning paradigms on such topologies. One of the great shortcoming of these architectures, however, is their problematical implementation in hardware where the high degree of interconnectivity required for all but trivial problems is simply not practically feasible. It is fair to say that the inherent parallelism which renders these architectures so powerful is rarely utilised. This represents something of a dilemma for the image processing community as large scale problems require prohibitively long training times where the neural network paradigm is implemented on a serial machine. LeCun et al *[60]* reports that training a standard backpropagation net on 7291 16 by 16 binary characters for an OCR system takes over 2 days on a Sun4 workstation. This is not a problem if training can be performed 'off-line' but obviously falls short of the criteria required for adaptive 'on-line' systems.

Bledsoe and Browning *[61]* introduce a pattern recognition technique which can be supported very neatly using standard RAM like memory components. Such architectures, often referred to as weightless or Logical Neural Networks, may be trained extremely rapidly and offer an attractive prospect for hardware realisable neural network architectures. Such a scheme has been employed by the WISARD *[62]* neural network which, to date, remains one of the few neural network architectures which can be trained on high resolution images (i.e. 0.25 million pixels) at video frame rates and certainly the only one which can do it at a modest cost. A typical WISARD system is shown in fig.5.1

5.3. The WISARD architecture.

LNN nodes consist of networks of relatively sparsely connected Boolean operators which are mapped onto a binary input retina. The precise functionality of each operator in the node is determined by a small sample of bits within the input pattern - each n-bit sample is called an n-tuple. These functions can be readily realised using standard VLSI RAM components so that every combination of binary elements within the n-tuple form an n-bit address to each location within the system memory. The memory requirements grow exponentially with tuple size and, as a consequence, these nets are typically sparsely connected with tuple sizes varying from between 2 and 8. The response from each function is summed in a discriminator to give an overall response from the pattern on the input retina.

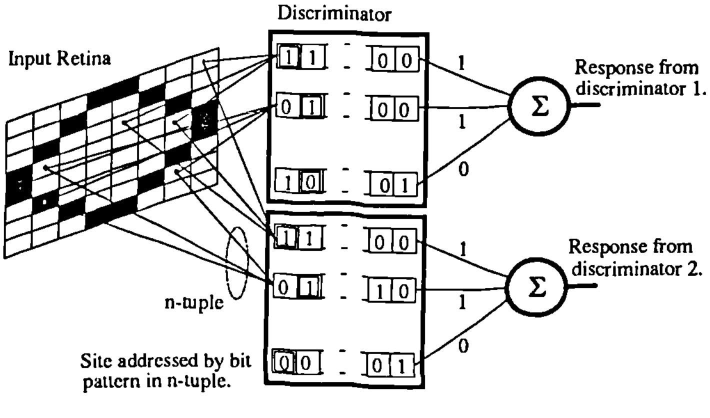

Discriminator
Input Retina
1
1
0
Response from discriminator 1.
Σ
1
1
0
Response from discriminator 2.
Σ
n-tuple
Site addressed by bit pattern in n-tuple.

Fig. 5.1 A typical WISARD system

Classically, the WISARD architecture is used as part of a supervised learning system where one discriminator is dedicated to each class of input pattern and training affects only the contents of the discriminator assigned to that class . Data is presented to the network during training and a '1' written to the sites addressed by the pattern within each tuple in the assigned discriminator. During the training phase a generalised representation of the prescribed class is built up within each discriminator.

After training has been completed, a pattern is presented to the network and the combination of binary elements within each tuple addresses a site in the system memory which may or may not have been filled during the training phase. The contents of the site addressed by each tuple within the discriminator is summed to form the response to that pattern. If the input pattern shares some of the same features as the class assigned to that discriminator during training then it will evoke a response from the system. The larger the extent of the shared features, the larger the response. If successfully trained, the system can give a high response to unseen data. It is this ability to generalise which gives these nets powerful recognition properties.

The tuple size and the assignment of the tuples to specific sites within the input retina can have a profound effect upon the behaviour of the network. As a prelude to the application of a WISARD type network to our particular problem it is worth investigating how the tuple size and the nature of the pattern domain is likely to affect the performance of the system.

5.4. How does the tuple size affect the performance of a Logical Neural Network ?

5.4.1. The nodal transformation of an n-tuple

An LNN node is not simply a digital implementation of a sum-of-weights node - its transfer function is more akin to the so-called 'sigma-pi' units described by Rumelhart *[48]* whose output is the weighted sum of the products of certain input lines. Such nodes can elicit higher order statistics from the input data than a basic sum-of-weights node and yields a distinctly non-linear type transfer function.

Gurney *[63]* gives the transfer function of a simple single 2-tuple node after training as:

$$y = s_{00}(1-x_1)(1-x_2) + s_{01}(1-x_1)x_2 + s_{10}x_1(1-x_2) + s_1x_1x_2 \qquad (24$$

Where:

y is the node output.

x_1 is value of binary input on 1st line of the tuple.
x_2 is value of binary input on 2nd line of the tuple.
s_{00} is the value at the site addressed when $x_1=0$ and $x_2=0$.
(set to 1 if pattern occurred during training).
s_{01} is the value at the site addressed when $x_1=0$ and $x_2=1$.
etc.

Equation (24 shows that an n-tuple node will extract n_{th} order statistics from the input data - the behaviour of the node becomes increasingly non-linear as the tuple size grows. It is generally true to say that for most pattern recognition tasks a non-linear node will extract more 'interesting' (and less trivial) features from a data set and it is instructive to appraise the potential of such nodal transformations in the context of our particular application.

5.4.2. Response profiles and tuple size - an analysis.

How does the response of a discriminator vary when presented with patterns not present in the training set?

Consider a D dimensional binary pattern, P, where every element within the pattern is assigned to an n bit tuple. There are D/n, 2^n bit memories within the discriminator. After training with a single presentation of P a '1' will be written to a single site within each memory (see fig.5.1). Presenting P to the trained net addresses each and every '1' set during training and the response of the node is D/n. Normalising the response gives:

$$y_0 = 1 \qquad (25$$

If pattern P_1, a Hamming distance of 1 from P, is presented to the network, one of the tuples will fail to address a filled site and the response of the discriminator drops by n/D. So that:

$$y_1 = 1 - {}^{n}\!/_{D} \qquad (26$$

Where:

y_1 is the normalised response of the node to a pattern P_1 which is 1 bit different from the training pattern P.
n is the tuple size.
D Is the dimension of pattern P (and P_1)

If pattern P_2 (a Hamming distance of 2 from P) is presented then, as before, the first bit difference will bring about a n/D change in the discriminator but the second bit difference may occur in the same tuple as the first (so the response remains 1-n/D) or in a different tuple in which case the response drops by a further n/D. The probability that the second bit difference occurs in a different tuple as the first is 1- n/D. Then:

$$y_2 = 1 - n/D - n/D(1 - n/D) \qquad (27$$

Where:

y_2 is the response of the node to a pattern P_2 which is 2 bits different from the training pattern P.

Generally, for very large D, node response to a pattern P_h which is Hamming distance h from training pattern P may be approximated by:

$$y_h \approx 1 - n/D \sum_{i=1}^{h} (1 - n/D)^{i-1} \qquad (28$$

Using the identity:

$$\sum_{i=1}^{h} a^{i-1} = \frac{1 - a^h}{1 - a} \qquad (29$$

Yields:

$$y_h \approx (1 - n/D)^h \qquad (30$$

This function is plotted below in fig. 5.2.

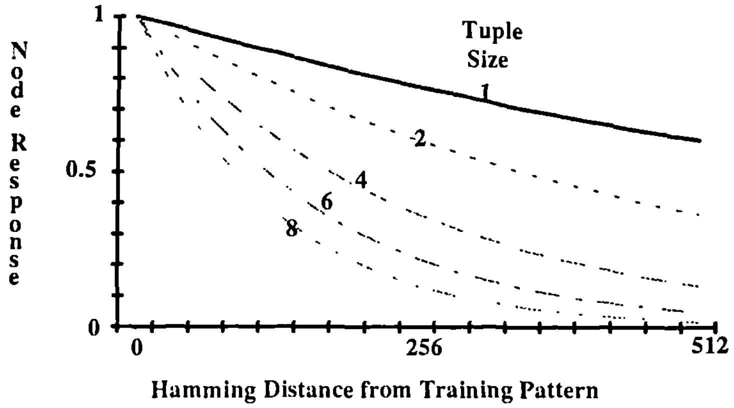


Fig. 5.2 Response profiles for n-tuple nodes

Fig. 5.2 gives an indication of the ability of the node to generalise. A low tuple size restricts the number of features that may be represented within a function and good generalisation is achieved through a coarse representation of the pattern feature space. As the tuple size grows, the feature space becomes increasingly fine-grained and the localised response of the node to images close-by in the pattern space, or its selectivity, becomes more pronounced. Thus, the generalisation ability of the net (the ability to respond to unseen inputs) is achieved at the expense of its selectivity (the ability to respond unambiguously to a given class). This trade off between selectivity and generalisation is dictated by the tuple size. It is problem specific and is typically resolved, to the chagrin of many researchers in the field, more by empirical means than by any set methodology (rather akin to the problem of selecting the number of hidden nodes in a backpropagation network, for example). Do the requirements of our coding scheme provide us with any clues to help select the optimum tuple size?

5.4.3. The contrasting roles of image classification and coding systems.

Consider the idealised response profiles of a classifier scheme for 2 classes $\{A_1,A_2,A_3\}$ and $\{B_1,B_2,B_3\}$ shown in fig.5.3. The objective of the system is to determine if the pattern presented to the net belongs to either of the training pattern classes. The output of the discriminator will respond with a '1' if the appropriate class is presented and a '0' otherwise.

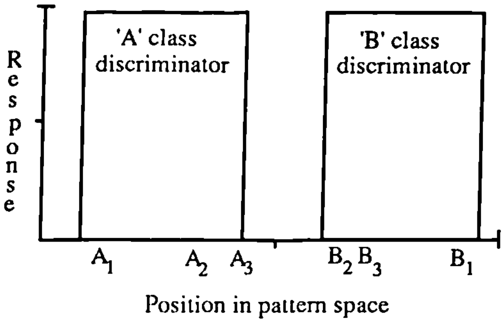


Fig.5.3 The idealised response of a 2 class classifier.

Suppose now that this net is to be used the basis of a coding scheme to indicate the relative positions of each of the patterns in the pattern space. The response of each discriminator forms an element of the code. The codes for the training data are thus:

$A_1 = [1,0]$
$A_2 = [1,0]$
$A_3 = [1,0]$

$B_1 = [0,1]$
$B_2 = [0,1]$
$B_3 = [0,1]$

This code does not furnish us with a great deal of information regarding the relative positions of the patterns in pattern space and it is apparent that the criteria for training nets to behave as classifiers is not particularly suitable for our coding requirements. A fundamental requirement of the coding scheme is that the coding transformation learnt by the node preserve the relative relationships between the patterns in the original domain (i.e. a conformal mapping). A node trained as an optimal classifier will not do this.

If we change the tuple size to 1 then the output of the node is linearly proportional to the hamming distance from the training pattern - this is shown in fig.5.3. A coding scheme based on this tuple size will better reflect the hamming distance between the patterns in the original space (whether or not this corresponds to a human beings notion of 'distance' in pattern space is quite another matter however).

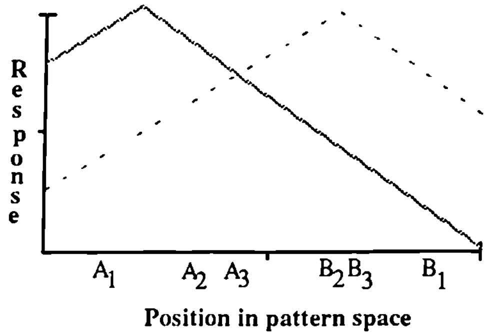


fig. 5.4 Response profile of a 2 class, 1-tuple system with one training example.

Such a scheme yields the following codes:

$A_1 = [0.92,0.35]$
$A_2 = [0.78,0.58]$
$A_3 = [0.6,0.72]$
$B_1 = [0.45,0.92]$
$B_2 = [0.43,0.97]$
$B_3 = [0.22,0.66]$

The Euclidean distance between the codes for each pattern better reflects their position in the original pattern space and is more suited to our requirements.

5.4.4. The effect of tuple size for a simple coding scheme.

The objective of the following test is to determine how the tuple size affects the performance of a coding mechanism where the code is taken to be the output from a single discriminator - i.e. it is scalar. The data set, is identical to that shown in fig. 3.9. The tuples are mapped randomly onto the complete retina (see 5.6) and the discriminator trained with one presentation of a single pattern, chosen from the data set. After training each pattern within the data set is presented to the net and the output of the discriminator normalised to floating point number where $0 \leq code \leq 1$. The test outlined in 4.2.3 was repeated for a LNN.

The discriminator was then cleared and trained with a different pattern from the data set and the results recorded. This procedure was repeated with each pattern as the training prototype and the best results shown below in fig. 5.5.

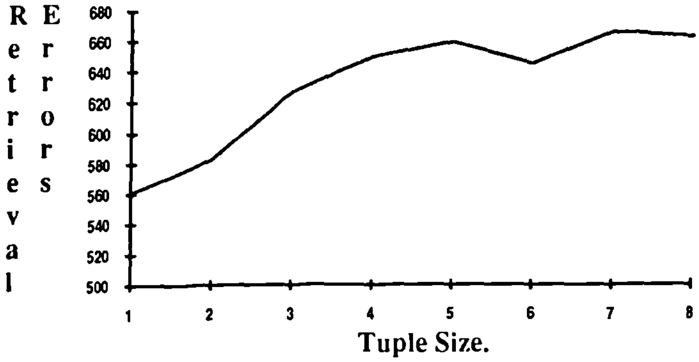


Fig. 5.5 Effect of tuple size on code performance.

These results suggest that the higher order nodal transformations produced by larger tuple sizes do not preserve the relative topological relationships between images in the pattern space and thus are not well suited to this simple scalar valued coding scheme. Fig.5.5 indicates that a sum-of-weights type linear network (or a trivial 1-tuple discriminator node which merely measures the Hamming distance between training pattern and target) will provide a better basis for the simple coding scheme described. This consolidates the hypothesis outlined in 4.4.3 which concluded that a linear sum-of-weights node might be more suited to this particular application than a non-linear one.

However, as the number of nodes increases, so that the code becomes a vector instead of a scalar, other factors come into play which warrant the use of larger tuple sizes. This is discussed in 5.7.

5.5. Training and System Performance

One of the great strengths of the neural network approach to pattern recognition over more deterministic methods lies in their ability to generalise through a learning phase. The neural net is able to do this because a generalised representation of the candidate class is built up within the node during training. Seen in this light one, could argue that the results reported in 5.4 were not representative of the capabilities of a neural network system as we considered a very trivial case - that is, a discriminator trained with one pattern. Such a discriminator could hardly be said to contain a generalised representation of the data set characteristics.

5.5.1. Training and tuple size - an analysis

The response a discriminator after training may be viewed as the superposition of individual profiles for each training pattern - this is shown in Figs. 5.6 and 5.7. The tuple size has a profound effect upon the shape of the profiles and consequently determines the extent of the training required by that node. Figs. 5.2 and 5.6 shows that the distinct response profiles associated with larger tuple sizes will restrict the area of influence of a discriminator in pattern space and thus will remove inter-class ambiguities. However, these sharp profiles render the scope of the class domain impractically small and this must be extended by careful training on several class prototypes which reflect the intra-class diversity.

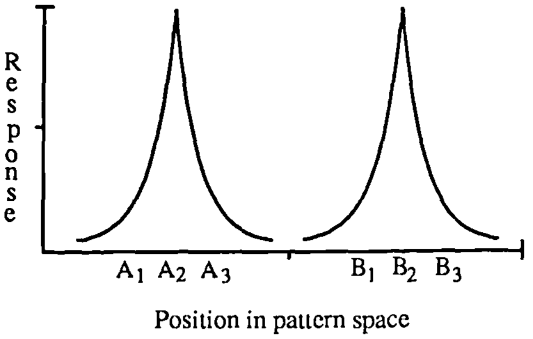


Fig. 5.6 2 class response profile with a single training example

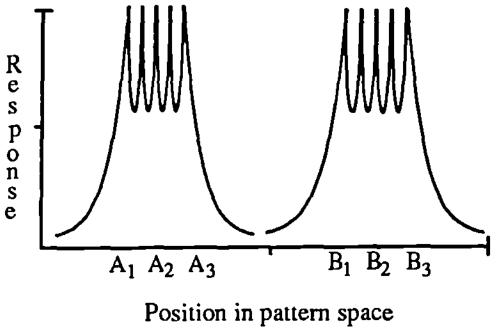


Fig. 5.7 2 class response profile with multiple training examples.

The sharp profiles produced by large tuple sizes can cause local anomalies in the response of that node (the 'wells' in the profile in fig. 5.7) and training for these systems must be more comprehensive than for small tuple sizes (where the 'wells' are shallower).

The reduced functionality of small tuple discriminators means that there are fewer sites to fill during training and such nets are prone to saturation where the discriminator exhibits a high response for the whole data set with a subsequent loss in selectivity.

Matching the extent of the training to the nature of the problem is typically solved through heuristics which poses something of a stumbling block for neural network practitioners and is especially difficult in applications where the notion of class membership is fuzzy, such as in unsupervised learning systems, for instance. A mathematical analysis of the training process may provide some insights that might help to resolve this approach.

Consider an ensemble of binary patterns with an average bit density γ - that is, there are on average $n\gamma$ '1's and $n(1-\gamma)$ '0's in each n bit tuple within the discriminator. During training, a '1' is written to the site addressed by the pattern in each tuple. The probability, P_A ,of site A being addressed within a function is given by:

$$P_A = \gamma^r (1-\gamma)^{n-r} \qquad (31$$

Where:

r is the number of '1's in the n bit address.

For a '1' to be written to an unfilled site after T training presentations 2 conditions must be satisfied:

i. The site was unaddressed during the previous T-1 training examples.

ii. The site is addressed by the T_{th} training pattern.

Thus, the number of sites filled in an n-tuple containing 2^n Boolean function after T training examples, S_T, is given by:

$$s_T = \sum_{m=0}^{2^n-1} \sum_{t=1}^{T} (1-P_m)^{t-1} P_m \qquad (32$$

Where:

P_m is the probability of the m_{th} site in the function being addressed.

For a 3 tuple:

$$P_0 = \gamma^0(1-\gamma)^3$$
$$P_1 = \gamma^1(1-\gamma)^2$$
etc

From the identity in equation (29, (32 becomes:

$$S_T = \sum_{m=0}^{2^n-1}\left(1-(1-P_m)^T\right) \qquad (33$$

So that:

$$S_T = 2^n - \sum_{m=0}^{2^n-1}(1-P_m)^T \qquad (34$$

Normalising gives:

$$S_T = 1 - \frac{1}{2^n}\sum_{m=0}^{2^n-1}(1-P_m)^T \qquad (35$$

This function is plotted in figs. 5.8 to 5.11 for various tuple sizes and image densities. These graphs confirm that low tuple sizes require less intensive training than higher tuple systems and are prone to saturation, especially where the average density of the data set approaches 0.5. It is evident that low density images warrant higher tuple sizes than high density images to achieve the same degree of generalisation after training.

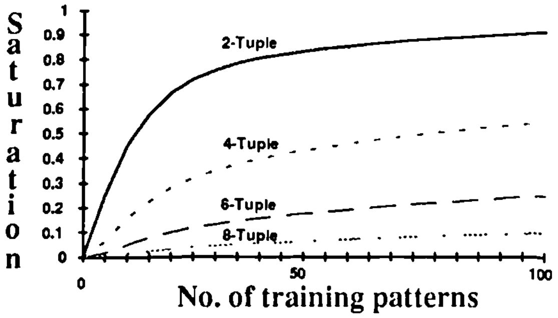


Fig. 5.8 Saturation vs training with average image density=0.1.

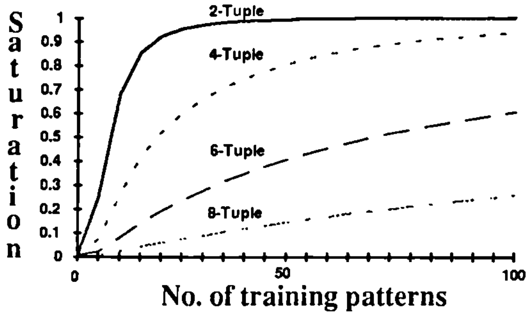


Fig. 5.9 Saturation vs training with average image density=0.3

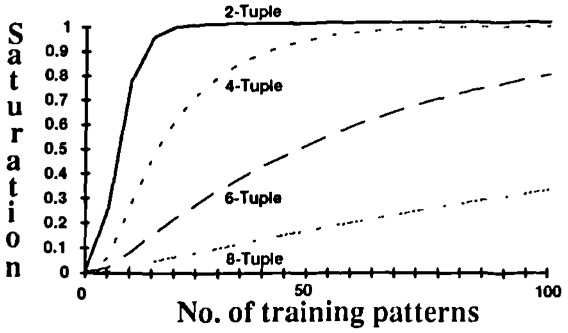


Fig. 5.10 Saturation vs training with average image density=0.5

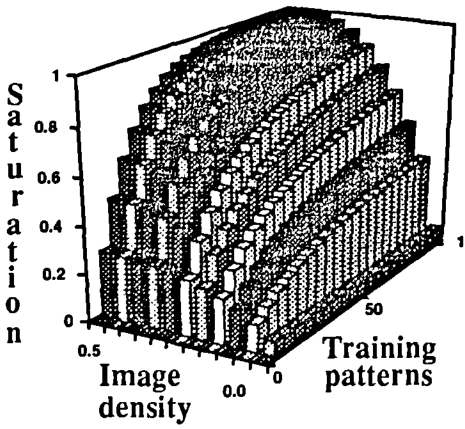


Fig. 5.11 Saturation vs training with a 4 tuple discriminator.

5.5.2. Training by image shifting for a practical coding scheme

We can see how training affects the performance of the coding transformation by repeating the experiment outlined in 5.4.4 but presenting several shifted versions of the pattern during the training phase. In this experiment the pattern is translated in the x and y planes before presenting it to the input retina - one complete training cycle for a pattern consists of the following translations:

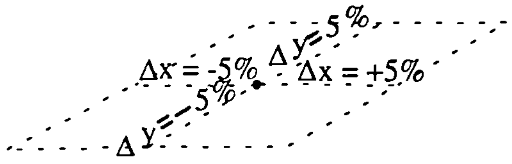


Fig. 5.12 Image shift during 1 training phase.

Patterns are truncated at the borders of the retina and the locations not occupied by the new pattern position are filled with '0's.

The performance of a retrieval mechanism based on this training regime is shown in fig. 5.13

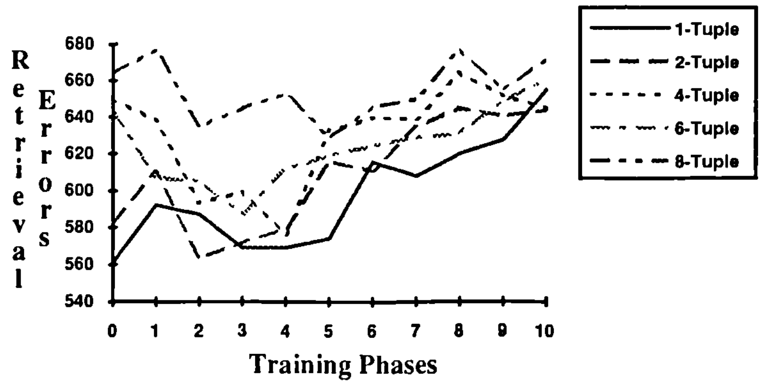


Fig. 5.13 Variation of retrieval performance with training intensity for shifted patterns with randomly mapped tuple sites.

Fig. 5.13 indicates that, for a single node system, training does not increase the efficacy of the coding mechanism and that no improvement can be made upon a simple 1-tuple system with a single training pattern presentation. This result provides more empirical backing to the hypothesis put forward in 5.4.3. It is interesting here to note that the training image which gave best retrieval performance for a given tuple size and training intensity (i.e. for the results given in fig. 5.13) did not remain constant but changed as the system parameters were altered (although recurring images tend to produce best results).

The standard deviation of the code produced by the scheme outlined above is shown below in fig. 5.14. this diagram suggests that the standard deviation of the code provides no hard and fast clues as to its performance and that choosing the training patterns to maximise the variance of the node output will not always result in an optimum coding scheme (although, generally, higher variances will tend produce better results). Each point corresponds to the best performance achieved for a given number of training passes and a particular tuple size.

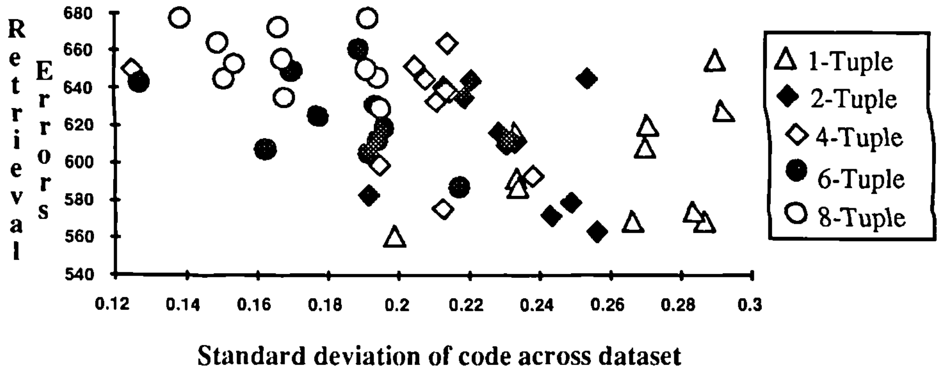


Fig. 5.14 Relationship between standard deviation of code and its performance for randomly mapped tuples.

5.5.3. Training by adding noise to an image for a practical coding scheme

In this experiment the discriminator is trained with several presentations of the same pattern where successive presentations corrupted with a fixed amount (4%) of random noise. This noise tends 'spread' the image in the feature space and produces a more generalised version of the image. This technique has been exploited by Aleksander *[64]* in his GRAMS with favourable results. Is it applicable to our system?

The best achievable results (after repeating the experiment with each pattern as a training prototype) are shown in fig. 5.15. This shows that generalising a pattern with noise spreading during training can produce better results than both training with shift and no training at all for a single node system but the improvement is quite marginal.

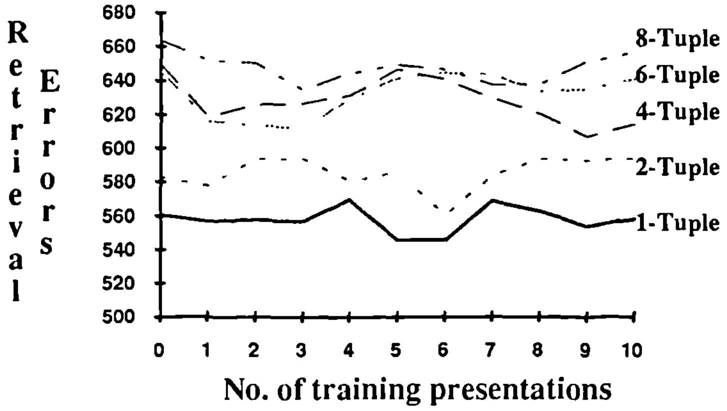


fig. 5.15 Variation of retrieval performance with training intensity for noisy patterns with randomly mapped tuples.

As with earlier experiments of this type lower tuple sizes give the best results. The standard deviation of the code produced by the scheme outlined above is shown in fig.5.16. (as in fig.5.14 each point corresponds to the best performance achieved for a given number of training passes and a particular tuple size).

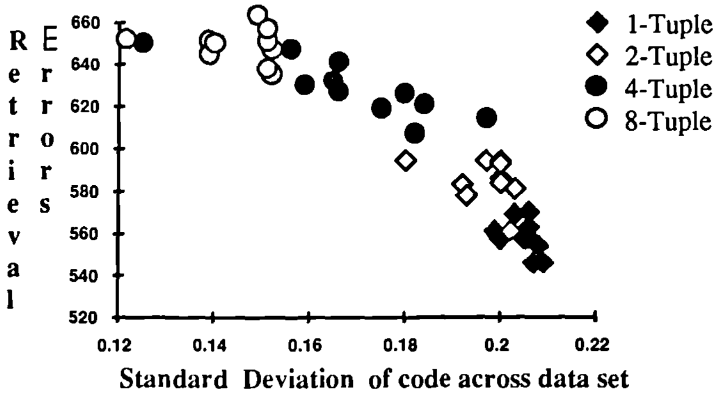


Fig. 5.16 Variation of retrieval performance with standard deviation of code for noisy training data.

Fig. 5.16 indicates that for noisy training patterns there is a distinct correlation between the standard deviation of the code and its performance - although the relationship is only a loose one. This provides a rough metric to help find the pattern and training schedule which optimises the performance of the code (note that this relationship does not hold so reliably for higher tuple sizes but this is academic as lower tuple sizes produce the best results anyway).

5.6. Mapping the Tuples onto the Image.

5.6.1. Why do some tuple mappings perform better than others?

On the classical WISARD system operated in a supervised training mode, mapping the tuples randomly onto the retina tends to produce the best results*[62][65]*. Why is this?

Meaningful real-world images contain many localised artefacts brought about by low luminance variance or continuous line segments across an image, for example. These artefacts mean that there is typically a high degree of localised image correlation within the data which will only be extracted if the tuples themselves are mapped locally onto the retina. It is possible to see the extent of localised correlation within the data set by observing the amount of information gleaned from various mapping strategies. This can be measured by recording the frequency of occurrence of tuple patterns for each tuple across the whole data set. The information, or entropy, H of each tuple is given by:

$$H = -\frac{1}{n}\sum_{i=0}^{2^n-1} P_i \ln_2 P_i \qquad (36$$

Where:

H = The average information per bit within the tuple.
P_i = Probability that site i is addressed by the tuple.
n = Tuple size.

A low entropy means that the tuple has extracted correlated pattern characteristics from the data. The average entropy per bit for local, random and maximally disparate tuple mappings for the data set given in 4.2 is shown below in fig.5.17. This indicates that the local mapping will elicit the higher order correlation inherent within local image artefacts.

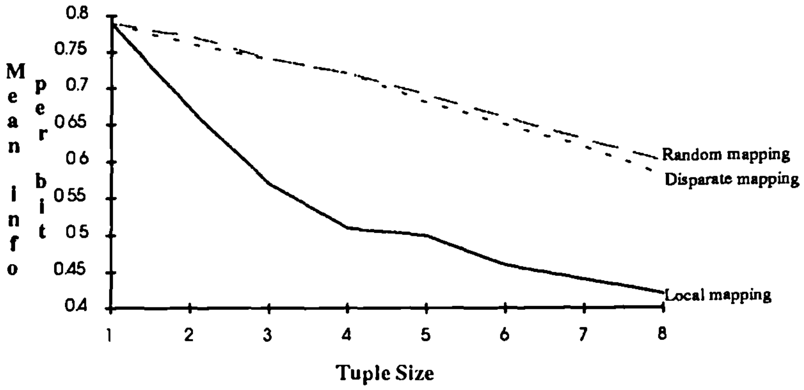
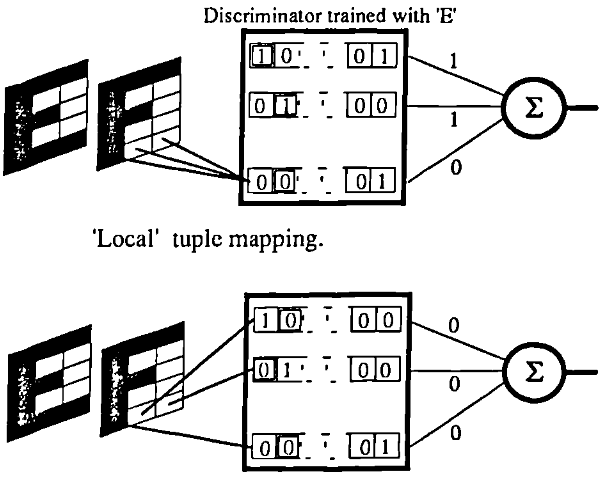


Fig. 5.17 Average entropy for mapping strategies for the data set shown in fig. 4.2.

Bearing in mind that the primary objective of most pattern recognition tasks is the extraction of highly correlated features, it seems, at first sight, counter-intuitive that the best results should be achieved by de-correlating these artefacts through random mapping of the tuples. Why should this be the case?

Consider a 3-tuple system designed to distinguish between the letters 'E' and 'F', shown in fig.5.18. In this example the discriminator has been trained with just a single presentation of the letter 'E'.



'Local' tuple mapping.

'Random' tuple mapping.

Fig. 5.18 The effect of mapping tuples in a simple discriminator system.

The only feature which enables us to distinguish between the two classes is the two right-most pixels in the last row of the retina. The object of the mapping should be to maximise the effect of this discriminant feature. The first mapping will concentrate the effect of this feature in just one tuple so that the response of the discriminator trained on 'E' to the letter 'F' is: $\text{Response}_{\text{max}}$-1. The second mapping scatters the effect of the distinguishing features amongst 3 tuples so that the response is now: $\text{Response}_{\text{max}}$-3. Thus, in this example a 'random' mapping will improve the differential response of the system.

Figs. 5.19 and 5.20 compare the results obtained with local and random tuple maps for the experiments outlined in 5.5.2 and 5.5.3 confirms this analysis.

As a corollary to this result, it is reasonable to suggest that the most effective mapping would be one where the local effects within an image are completely de-correlated (i.e. they influence as many tuples as possible). Such a mapping may be achieved by maximising the Euclidean distance between the sites assigned to a tuple across the retina. Fig. 5.21 demonstrates the performance of a code resulting from such a mapping and shows an improvement over random mapping.

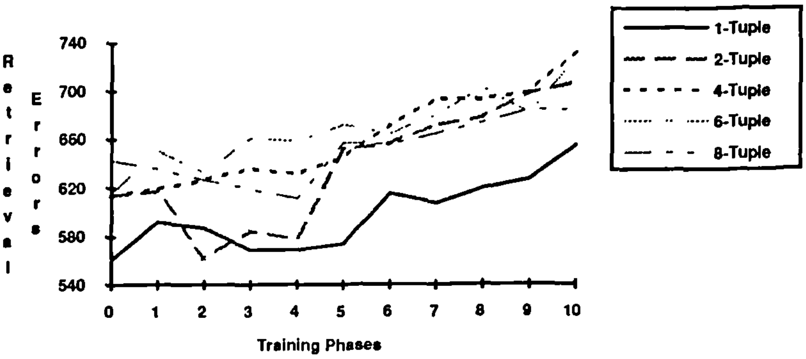


Fig. 5.19 Retrieval performance for locally mapped tuples and shift training.

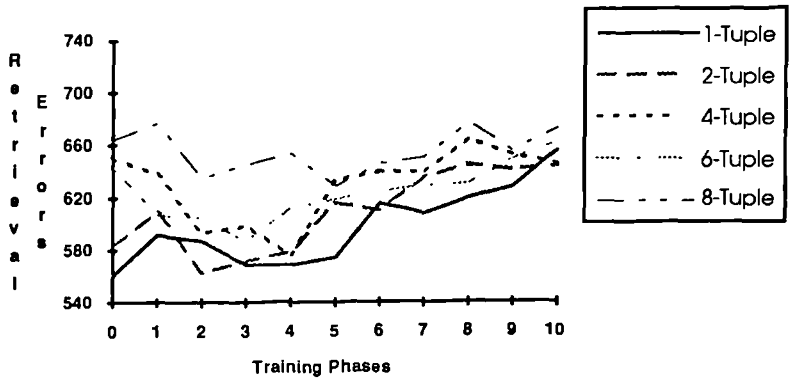


Fig. 5.20 Retrieval performance for randomly mapped tuples and shift training.

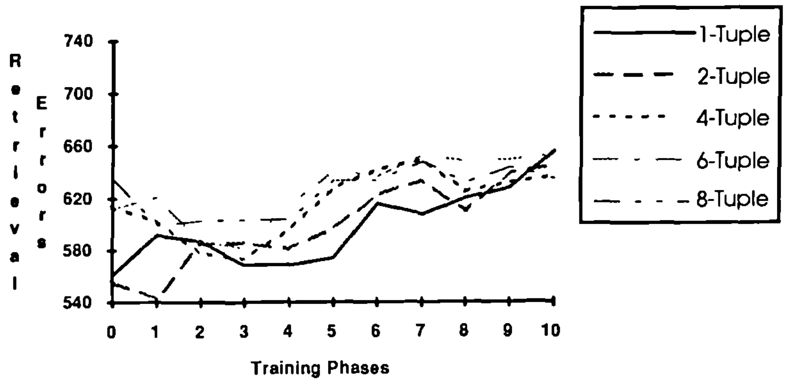


Fig. 5.21 Retrieval performance for disparately mapped tuples and shift training.

It is worth nothing that, apart from a 2 tuple system with 1 or 2 training passes (according to the mapping used), a single tuple system will still outperform higher tuple ones.

5.6.2. The tuple as a feature extractor - what's going on ?

The question we hope to address here is as follows: In a classical WISARD system do large tuple systems outperform small tuple ones simply because they extract higher order features from the data set? On the face of it this seems a reasonable assumption because the role of any pattern recognition system is to represent highly correlated features in a smaller and so more manageable domain. Since large tuple sizes can indeed extract higher order statistics from the data set it is natural to assume that this property is responsible for the improved performance. However, we have already seen that it is necessary to de-correlate local features by appropriate mapping of the tuples for the system to perform well and this raises an interesting dichotomy.

To help resolve this we offer the following conjecture: *In a classical WISARD implementation, large tuple sizes are useful not because they extract higher correlations from the data but because they offer improved prospects for inter-class orthogonality.* This is demonstrated in fig. 5.22. This improved separation between classes comes about because the feature space within the discriminator is a vastly expanded version of the original pattern domain - the size of this domain grows exponentially with tuple size. This is often referred to as $[0,1]^n$ space and an excellent review of its properties is given by Kanerva *[66]*.

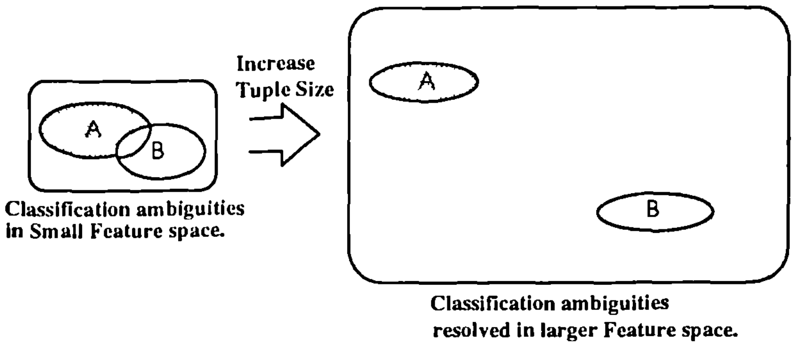

Increase
Tuple Size
A
B
Classification ambiguities
in Small Feature space.
A
B
Classification ambiguities
resolved in larger Feature space.

Fig. 5.22 Inter-class orthogonality is improved in larger feature space.

Kanerva shows that, as the tuple size grows, most of the features that can be represented within that tuple become maximally orthogonal to all other features in the features space. This means that there is improved separation between inter-class candidates; though this is achieved at the expense of generalisation (the intra-class candidate separation).

The high degree of correlation between local image artefacts in real world images means that much of the $[0,1]^n$ space is un-addressed which in turn reduces the potential inter-class orthogonality that may be achieved by the recognition system. The full $[0,1]^n$ space can only be utilised if the entropy of the data set is 1. For a tuple with mean entropy H per bit across the data set the actual size of the $[0,1]^n$ space reduces from 2^n to $2^{H.n}$ (where $H \leq 1$). By mapping the tuples randomly onto the retina we can reduce these localised correlations, thus increasing H with a commensurate improvement in performance.

5.6.3. Mapping tuples to maximise the discriminant information.

Where images contain a large amount of common data (such as a large expanse of white background, for example), the relative performance of a system which monitors all of the retina will be degraded. This is because areas of the image which provide no class discriminant information are given the same emphasis as areas which do. The requirement to maximise the effect of the discriminant information has been discussed by Tattersall *[67] et al.*

An indication of the potential discriminant information within a single element on the input retina can be found by measuring its entropy, H, across the data set. If the pattern 'seen' by the tuple remains fixed across the whole data set then $H=0$ and it contributes nothing to the classification process. It is worth noting, however, that the entropy is just a first order measurement and does not furnish us with the information potential *between* elements on the retina (frequently called relative entropy *[68]*). In view of the fact that an image is characterised by a set of correlated elements, it is not surprising that the entropy only gives us an upper-bound as to the amount of information contained within the retina element (the reader will recall that the PCA transform produced by the sum-of-weights nodes discussed in chapter 3 took both spatial and ensemble correlations into account).

Filtering out low entropy sites within the retina forces the system to regard only those sites which contain highly discriminant information during the coding transformation. The experiment outlined in section 5.5.3 is repeated here with a range of entropy filtered masks. The results are shown below in figs. 5.23, 5.24 and 5.25.

These results show that the performance of the system can be improved if areas which contain no discriminant features are filtered out prior to the coding process. However, if this filtering is too severe then areas of fine detail are lost and the efficacy of the code degraded. It is curious to note that training with noise improves the performance of small tuple systems with a high degree of entropy filtering yet degrades it where the filtering is relatively light.

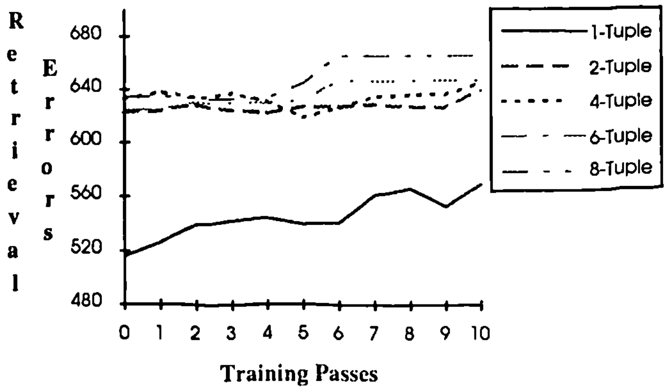


Fig. 5.23 Retrieval performance with noise training and H>0.25 entropy filter.

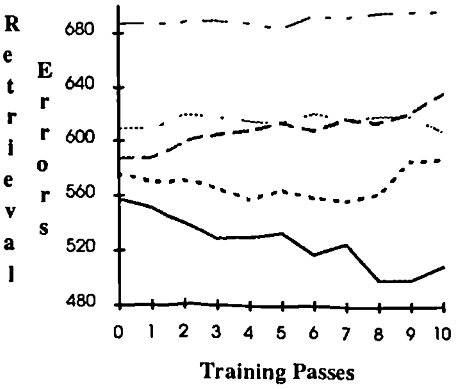


Fig. 5.24 Retrieval performance with noise training and H>0.5 entropy filter.

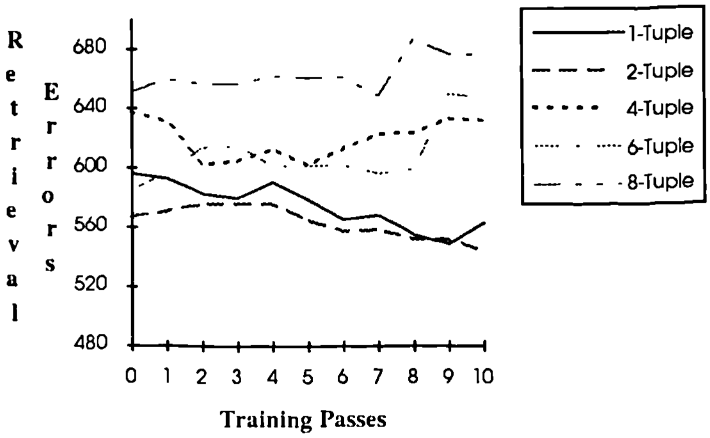


Fig. 5.25 Retrieval performance with noise training and H>0.75 entropy filter.

5.7. A Multi-Node encoding Scheme using Logical Neural Networks

The results presented so far in this chapter help to underpin some of the general principles governing the performance of a simple coding scheme using a single LNN node. Such a scheme represents each image as a scalar and it is, perhaps, a little hopeful to expect the vast amount of information contained within an image to be adequately conveyed by such a condensed representation. An important point to bear in mind at this juncture is that the role of the code is not to represent the *absolute* information within an image but rather the *relative* information that allows one image to be discriminated from the next.

The concepts governing the relationship between the number of nodes and the performance of the code have already been touched upon in chapter 4. It is helpful now to frame these in the context of an LNN system.

5.7.1. Matching the size of the code to the diversity of the database

The number of nodes required by the system is dependent upon the diversity of images within the database. A database of constrained image types requires fewer nodes than an unconstrained one to achieve the same level of performance. The following tests show how the performance of the code is affected by the diversity of the database. In the first test the database consists of just the first 4 classes of the data set shown in fig. 4.2 (that is, 'A' to 'D') with the number of examples of each class varied to investigate how the size of a database of fixed diversity (that is, just 4 classes) alters the efficacy of the retrieval mechanism. The discriminator was trained with one presentation of an image from within the database - each image was selected as a training candidate and the one which yielded fewest errors chosen for the test. The relative number of errors is defined as the actual number of retrieval mismatches normalised by the maximum possible number of errors (see 4.2.6). The results, shown below in fig. 5.26, represent the best possible performance of the system.

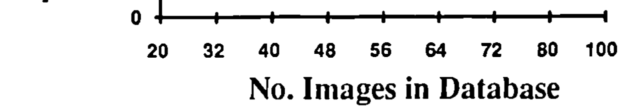


Fig 5.26 Retrieval performance for a database of fixed diversity.

The object of the next test is to determine how the diversity of the database affects the retrieval performance. The procedure outlined above was repeated but with the number of examples per class fixed (at 10) and the number of classes varied (from 4 to 10). The results are shown below in fig. 5.27.

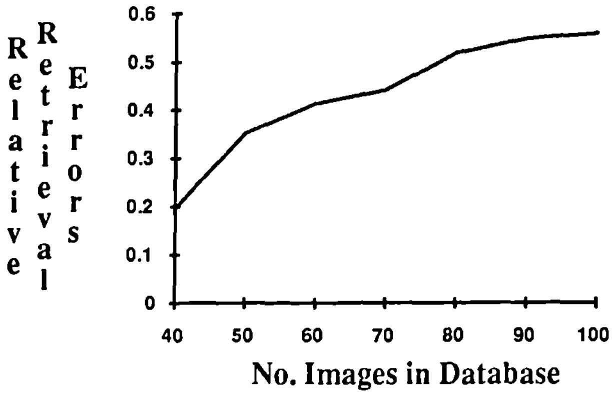


Fig 5.27 Effect of database diversity on retrieval performance

The object of the first test is to show that, for a database of fixed image types, the retrieval mechanism is relatively insensitive to its size. This helps to frame the results presented in fig. 5.27. This indicates that it is the diversity of image types and not necessarily the number of images within the database which impairs the performance of the code (although, of course, these two factors are linked to a certain extent for a practical database).

Assuming, then, that the database contains a 'theme' or set of 'themes' what is the most economic code that can be used? The following tests show how the retrieval mechanism improves as the number of nodes is increased for a range of randomly mapped tuple sizes. Apart from using multiple nodes instead of just a single node the tests are identical to those outlined in 5.4.4. In this case, each node is trained with a single presentation of one image selected from within the database. The retrieval errors are recorded and the discriminator cleared and re-trained. This procedure is repeated for each and every image in the database and the best results presented below in fig. 5.28.

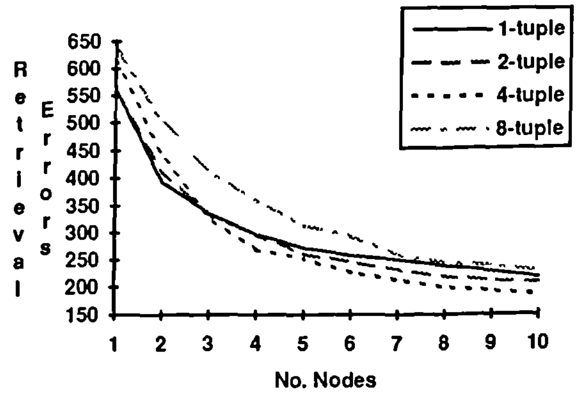


Fig. 5.28 Variation of retrieval performance with number of nodes for the database shown in fig.4.2. (no shift training).

Fig. 5.28 shows that, for systems containing more than a single node, larger tuple sizes will tend to improve the performance of the code. This is because, in the higher dimensional $[0,1]^n$ space produced by large tuple sizes, it is easier to select orthogonal training candidates for successive nodes. This was discussed in 5.6.2. Thus, although the expanded feature space can disrupt the conformal mapping required by our code, it does enable more information to be packed in the code because successive code elements are un-correlated. At some stage the benefits afforded by this expanded feature space are offset by the breakdown in the conformal mapping incurred by the non-linearity of large tuple sizes (see 5.4.3). For this particular data set a 4-tuple gives optimum performance.

5.7.2. The effect of training in a multi-node coding scheme

In 5.6.2 we showed that, although training did indeed improve the performance of a single node system for larger tuple sizes, in general, it still could not improve upon that for a 1-tuple system without training (the exception being a 2-tuple system with a single shift train phase, shown in fig. 5.21). However, since the size of the feature space becomes more important than the conformality of the transform for multi-node systems we can see that training is likely to improve the performance of such systems.

The experiments outlined in 5.5.2, which demonstrated the effect of training in a single node system, are repeated now for a multi-node scheme and the results presented in figs. 5.29, 5.30, 5.31 and 5.32 for 1, 2, 4 and 8 tuples respectively. These confirm that training the nodes by shifting the training exemplar will enhance the performance of multi-node systems, provided that a 1-tuple system with less than 3 nodes is not used. It is interesting to note that all of these systems (barring the case mentioned earlier) exhibit optimum performance when trained with either 2 or 3 passes of the data set.

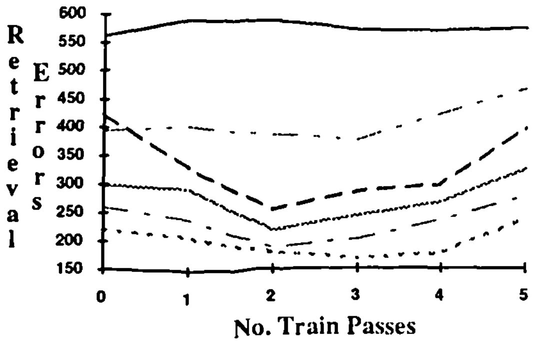


Fig. 5.29 Effect of shift training on a 1-tuple multi-node system.

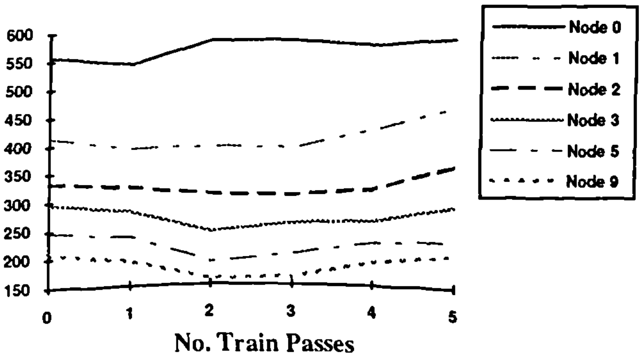


Fig. 5.30 Effect of shift training on a 2-tuple multi-node system.

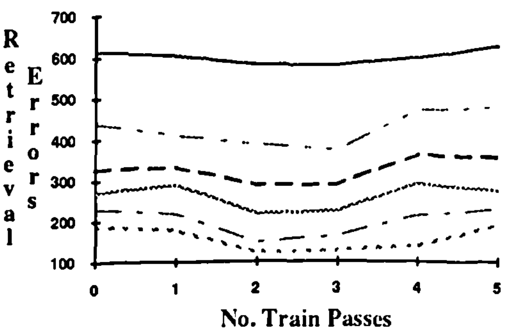


Fig. 5.31 Effect of shift training on 4-tuple multi-node system

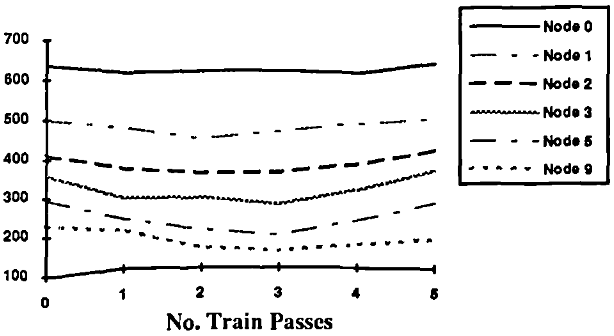


Fig. 5.32 Effect of shift training on 8-tuple multi-node system.

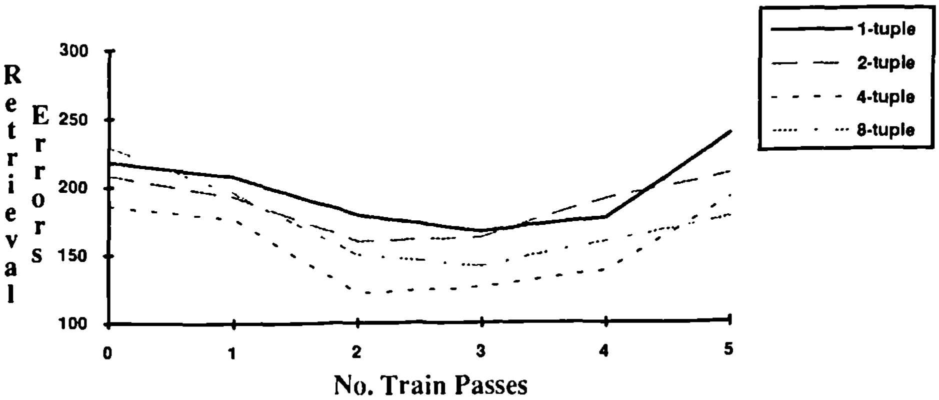


Fig. 5.33 Effect of shift training for a 10 node system

Fig. 5.33. shows that for a 10 node system a 4-tuple scheme with 2 shift training cycles will give the best retrieval performance for this particular data set. By way of comparison, the performance achieved with noise training for a multi-node scheme are given in fig. 5.34. Shift training seems to produce better results than training with noisy exemplars - this is probably because such a training strategy imparts a degree of translational invariance into the system.

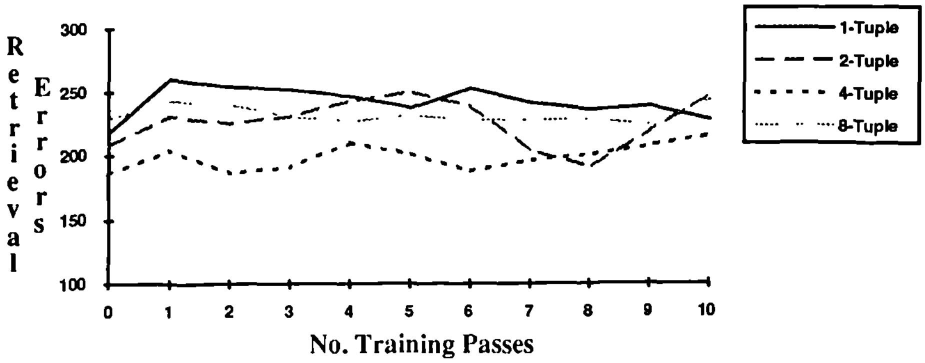


Fig. 5.34 Effect of noise training for a 10 node system (4% noise added per training pass)

5.8. An unsupervised Coding Scheme using a Logical Neural Network

The results presented thus far have shown the best possible results that may be achieved by the system. These have been found by training all possible patterns into the network and selecting those results which yielded fewest errors. In a practical environment the optimum training configuration for each node cannot be found by trial and error as the database is unlikely to contain a data set structure that permits retrieval 'errors' to be so easily defined.

Figs. 5.29 to 5.32 demonstrated that for a fixed number of nodes there is a certain degree of training which tends to improve results after which the performance begins to drop off. How can we determine the optimum extent of training?

Fig. 5.35 shows that there is a loose correlation between the standard deviation of the first node and the retrieval errors induced by the code for a given tuple size with shift training (as in fig.5.14 each point corresponds to the best performance achieved for a given number of training passes and a particular tuple size). A suitable regime might therefore be to continue the training phase until the standard deviation of the first node begins to drop off.

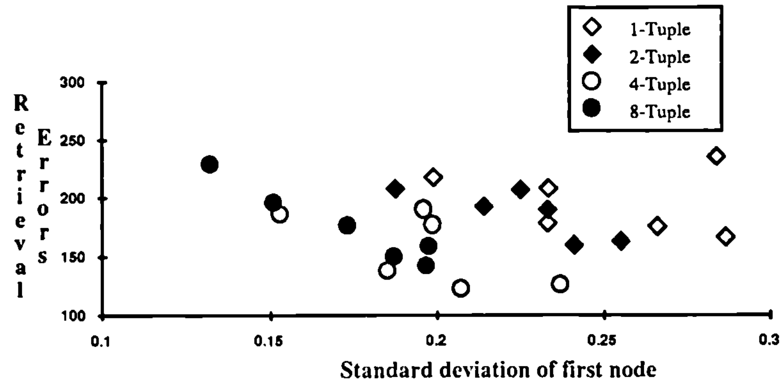


Fig. 5.35 Relationship between standard deviation of first code element and retrieval performance for a 10 element code with shift training.

This technique can also be used to select the most appropriate training pattern - deemed to be that pattern which, when trained into the discriminator, produces the highest variance. This does not quite result in optimum performance (compare figs. 5.33. and 5.39) but does allow us to go some way towards an automated system.

However, it is apparent from fig. 5.35 that the variance of the node cannot be used to select the optimum tuple size. This highlights an important and enigmatic issue common to LNN research, namely; 'How can we match the tuple size to the nature of problem domain ?' At present, to the chagrin of this researcher, this can only be resolved through empirical observation.

We have shown already that, for a multi-node system, larger tuple sizes will improve the efficacy of the retrieval mechanism. This is because the expanded feature space makes it easier to select un-correlated training candidates for successive nodes. However, after selecting that image which gives the best standard deviation for the first node, simply selecting that image for training into the second node which is least correlated with the first is not sufficient. This image must be both maximally un-correlated with the first yet still produce a high standard deviation across the data set.

The following automatic training scheme is proposed:

i. Train each pattern from within the data base into the first node. Training consists of a single training presentation of each pattern.

ii. Present each pattern from the database to the trained node. Record the standard deviation of each training candidate.

iii. Select that pattern which gave the largest standard deviation and train the node as before.

iv. Shift the pattern (as described in 5.5.2), train the node and record the standard deviation of the node across the whole data set.

v. Continue training until the standard deviation begins to drop off. Record the optimum number of training passes and re-train the first node.

vi. Repeat steps i. to iii. for subsequent nodes (ignoring any patterns already trained into the previous nodes).

vii. Measure the RMS correlation of the node to earlier nodes. Where:

$$Corrn_{ab} = \frac{\sum_{i=1}^{\substack{AllPatterns \\ inDatabase}} (op_{ai} - \overline{op_a})(op_{bi} - \overline{op_b})}{\sigma_a \sigma_b} \qquad (37$$

$Corrn_{ab}$	=	Correlation between nodes a and b.
σ_a	=	Standard deviation of node a.
σ_b	=	Standard deviation of node b.
op_{ai}	=	Response of node a to image i.
op_{bi}	=	Response of node b to image i.
$\overline{op_a}$	=	Mean response of node a.
$\overline{op_b}$	=	Mean response of node b.

and:

$$\text{RMS Corrn}_a = \sqrt{\frac{\sum_{j=1}^{a} Corrn^2{}_{aj}}{a}}$$

viii. Select training pattern which has highest standard deviation yet is *least* correlated with patterns trained in previous nodes. That is:

$(1\text{-RMS Corrn}_a) * \text{Std. Dev.}_a$ is maximum.

ix. Train this pattern into the node as in iv. to v.

x. Continue for each and every node.

Figs. 5.36 to 5.40 show the results for this training strategy for a range of shift training phases and tuple sizes. These results show that as training increases, a coding scheme with large tuple sizes will outperform one with small tuple sizes. This is because the increased feature space associated with large tuple sizes renders the discriminator less prone to saturation - this was discussed in 5.5.1.

It is worth noting that although a 10 node, 8-tuple system with 3 shift training phases will yield the best response, a 6 node, 4-tuple system with 1 shift training phase performs almost as well with less than 4% of the memory requirements. These results also show a curious consistency for the performance to plateau off somewhat for systems using more than 6 nodes. This seems to suggest that the data set can be represented adequately by a set of 6 prototypes (where each prototype is assigned to a node) with all subsequent prototypes providing little in the way of additional descriptor information. This notion appears to hold true irrespective of training intensity or tuple size. It is interesting to note that this effect was also observed for the sum-of-weights node, discussed in 4.3 which suggests that both the Logical and sum-of-weights type neural networks are performing similar operations.

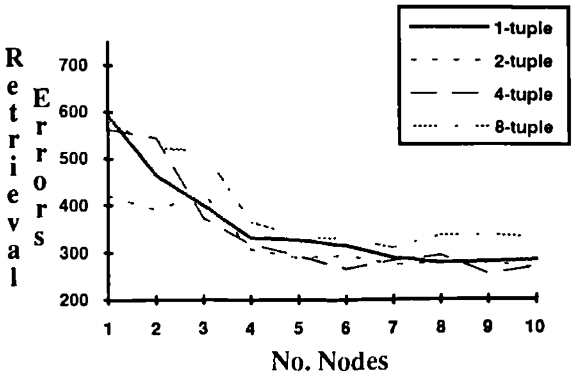


Fig 5.36 Retrieval performance for scheme outlined in section 5.8 with no training.

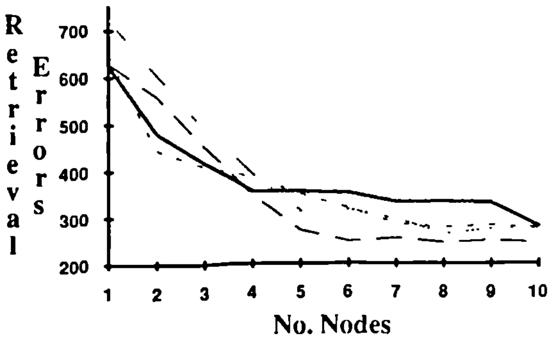


Fig 5.37 Retrieval performance with 1 shift training phase.

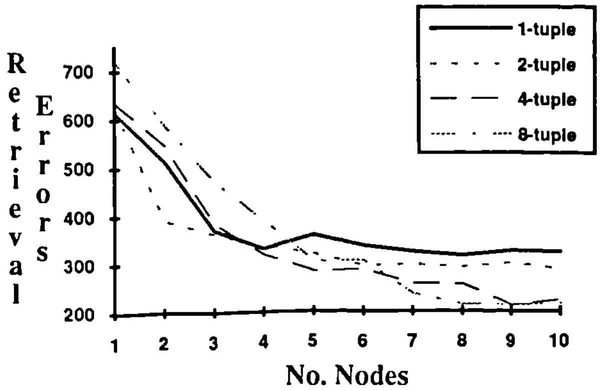


Fig 5.38 Retrieval performance with 2 shift training phases.

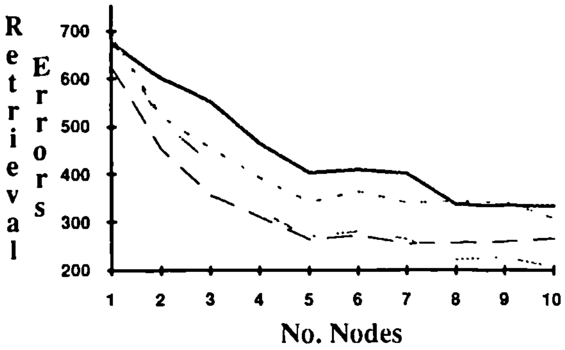


Fig 5.39 Retrieval performance with 3 shift training phases.

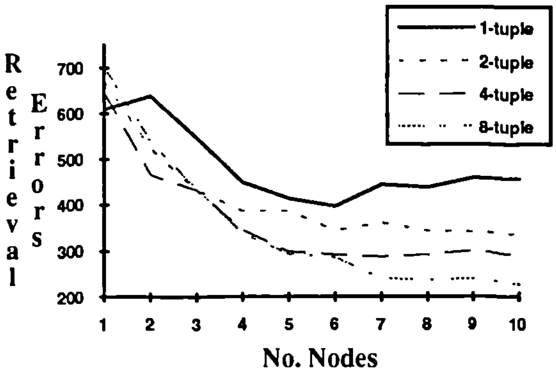


Fig 5.40 Retrieval performance with 4 shift training phases.

5.9. Discussion

Throughout chapter we have shown how a simple single layer Logical Neural Network can be used as the basis of a coding scheme for an image retrieval mechanism.

We have shown that the requirements of a coding scheme contrast markedly to those of a classification system. Section 5.4.3 highlights this point and reaches the somewhat surprising conclusion that, for a single node system, a 1-tuple discriminator will outperform larger tuple systems. This is because one of the main criteria for our coding scheme is that it preserve the relative relationships between the patterns in the original domain - that is, that nodal transformation of the neural network must be conformal. Larger tuple sizes exhibit increasingly non-linear transfer characteristics and will not produce this conformal mapping. Section 5.4 provides some mathematical and empirical evidence to corroborate this hypothesis.

Increasing the number of nodes offers the prospect of improved code performance. The economy of this, and indeed any code, is increased if the elements within the code are un-correlated. Because the feature space for small tuple systems is considerably smaller than for larger ones (the feature space grows exponentially with tuple size) it becomes increasingly difficult for the domains occupied by the training archetypes within the feature space not to encroach upon each other. The upshot of this is that successive code elements (from the output of the nodes) become increasingly correlated. This severely degrades the performance of the code. As the number of nodes is increased, the expanded feature space afforded by large tuple sizes becomes more important than the conformal mapping produced by the smaller ones - this is demonstrated in sections 5.6.2 and 5.7

Section 5.6 demonstrated that an LNN will perform best if the tuples are mapped to completely un-correlated areas within the image retina. Typically, the tuples are mapped randomly onto the retina to help de-correlate localised image artefacts. However, in 5.6.1 we presented a novel mapping strategy which can improve upon this technique. Here the tuples are mapped onto the retina such that the mean Euclidean distance between the tuple sites is maximised. This technique is applicable to any LNN system and is particularly beneficial for low-dimensional images. (e.g. the data set shown in fig. 4.2).

Unlike other neural network architectures, WISARD type networks only require one pass of the data set during the training phase. This is frequently referred to as 'one-shot' training. Whilst this affords us with some obvious speed advantages during training it does mean that every pattern exerts the same influence on the nodal transformation regardless of its context to the data set. This tends to render the learning process in WISARD systems as somewhat brutal since worthless features are given the same weight as important ones which is obviously going to degrade its performance. This is not the case with more conventional neural network models where the learning rate tends to spread the effect of a pattern across a wide section of the data set so that a kind of 'relative' learning takes place. This is in marked contrast to the kind of 'absolute' learning found in WISARD systems.

Some efforts have been made to circumvent this problem: Tattersall *et al [67]* biases the learning in favour of areas of high information content within the retina. Kerrin *[69]* has regarded the learning potential of a single tuple as a fixed resource so that frequently occurring features are learned at the expense of less important ones during a normalising phase.

We can conclude from this that WISARD systems come to the fore when training times are the prime concern and where the nature of the training data is such that only one pass is possible (such as in 'real time' training, for instance).

In our particular application, the data base cannot be regarded as a volatile data set since we can gain access to every pattern relatively easily during the training phase. Additionally, since 'off-line' training is unlikely to incur too many operational difficulties, a more refined training methodology using multiple presentations of the data set might perhaps prove more effective than simple 'one-shot' learning for this type of application. This issue is taken up in chapter 6.

In 5.5.1 we presented an analysis which helped to explain why large tuple systems require more extensive training than those with small tuple sizes. Section 5.5.2 indicated that there was a degree of correlation between the standard deviation of the code and the best level of training required by that node. The correlation is a loose one (and not really applicable to 1-tuple systems) but it does provide us with a rough metric with which to judge the extent of training most suited to a particular data set and topology. Additionally, the standard deviation of the code can be used to select the most appropriate training archetype (chosen from the data set) for any particular discriminator.

We also show that for a multi-node system working on the data set given in fig.4.2, training by shifting the image on the retina will outperform training with noisy exemplars.

In section 5.8 we combined all of the insights gleaned from this chapter and applied them to an automatic coding scheme for a simple database of machine printed characters. For this data set a 10 node, 8-tuple system with 3 shift training passes gave the best performance. The results achieved using these parameters are remarkably similar to the results obtained with the sum-of weights node in Section 3.5.2 of this thesis. The performance of these two systems, shown in figs. 3.19 and 5.38, indicate that, after training, there is little to choose between the two techniques. This suggests that the sum-of-weights nodes and LNN nodes are performing the same underlying processes. The training strategy here was completely automatic but in 5.7.2 we experimented with each every image as possible training exemplar in an attempt to get some idea of the best possible performance from the system. The results achieved using this exhaustive search method (which cannot be implemented for a practical system) were better than any technique used throughout the rest of this thesis. This shows that the performance of these systems is sub-optimal. We touched upon this in 3.3.5 where we concluded that coding transformations of these type required by our scheme will only produce optimal transforms where the distribution of the patterns is Gaussian.

The training strategy in the LNN implementation is a little crude as each pattern in the data base must be trained into all of the discriminators and the entire data set presented to each discriminator. Thus, the processing time grows linearly with the number of nodes and as a square of the size of the database. However, because they can be easily be implemented in hardware LNN's can be trained extremely rapidly and this shortcoming is not as serious as it would be for other neural network models. Nevertheless, these drawbacks suggest that it would perhaps be worthwhile investigating some other paradigms for Self Organising Logical Neural Networks to see if these processing overheads could be significantly reduced. This is the objective of the work described in chapter 6.

Chapter 6

A Novel Self Organising Learning Law for Image Coding.

6.1. Introduction

In this chapter we trace the development of a completely novel learning paradigm for a Self Organising neural network. The learning law began life as an adjunct to the work on LNN's described in chapter 5 of *this thesis.*

We give a brief overview of existing Self Organising learning paradigms for LNN's and assess their suitability for our own particular application. We take an empirical look at the underlying principles governing Self Organisation in LNN's and discuss how these might be implemented using a learning law. We go on to develop a mathematical formalism for this law and prove that the dynamic system governed by such a law will converge to the Principal Components of the data set.

Closer inspection of the form of the learning law reveals that it need not be restricted to just LNN's. We develop a form of the learning law which can be supported on a sum-of-weights type node and contrast the performance of this implementation with a LNN.

We set out to determine whether Self Organising LNN (SOLNN) architectures are more appropriate for our particular application than their sum-of-weights type counterparts, both in terms of speed of training, memory requirements and optimality of solution. We question the validity of LNN's for our particular application since the very attributes which render LNN's attractive, namely, speed of training for supervised learning systems and non-linearities that can be put to good use in classifier systems, are no longer applicable to unsupervised dimensionality reducing networks. We point out that LNN's are slightly more restrictive than their analogue counterparts and not particularly suited to our application. We compare the performance of three different Self-Organising neural network coding schemes for our image retrieval system.

6.2. An Overview of Self -Organisation in Logical Neural Networks.

LNN's are trained, typically, through supervised learning and comparatively limited research has been carried out on their application to Self Organising systems*[69][70][71][72][79]*. So that we might frame the work presented here in a better light, it is useful to review a cross-section of the existing SOLNN architectures and discuss their relative merits with respect to dimensionality reduction and image coding.

6.2.1. The Feature Sensitive Node.

Kerin *[69][73]* presents a Single Layer SOLNN, based around so-called Feature Sensitive Nodes (FSN's) which bear many similarities to Grosburg's ART network *[74]*. This architecture consists of an array of discriminator type nodes which attempt to group the input data into notional clusters.

The discriminator is similar to that presented in 5.3 except that the sites addressed by each tuple can take on positive or negative integer values. These sites are initially zero and incremented or decremented during training. If a node has been selected for training on a particular pattern then the site addressed by each n-tuple is incremented by an amount S and the remaining 2^n-1 sites are decremented by an amount W. The relationship between S and W sets the relative importance of the appearance or non-appearance of features (represented by a particular n-bit pattern on the n-tuple) in classifying the pattern as belonging to a particular class.

The FSN's respond to the frequency of occurrence of features within the patterns seen during training and compete for learning stimulus using a winner-take-all strategy where only the discriminator with the largest output undertakes any learning. The output of the node is defined as the sum of the thresholded sites addressed by the pattern at the input. The site output is thresholded to 0 if the site value is less than zero else it is thresholded to 1. Sometimes, according the to ratio of S/W the contents of each site across all tuple addresses is less than zero. Under these conditions the node will not contribute to the output of the FSN under any condition and is referred to as a 'dead' node. Dead nodes are discounted from the response of the FSN. The output from the FSN is normalised with respect to the number of live (i.e. not 'dead') nodes.

The winner-take-all strategy is further enhanced by ensuring that learning is only activated if the pattern has a high correspondence to an existing class (where one node is assigned to each class). Patterns which fail to score enough to be assigned to any of the existing nodes are assigned to a new node within the network. The threshold which determines this level of correspondence is set by the user. In this respect the learning algorithm is very similar to the Leader Algorithm used in classical clustering analysis.

Because this method is, essentially, a clustering algorithm, it is not particularly well suited to our type of application. This has already been discussed in section 4.5 and 5.4.3. In any event, the system classifies patterns according to the frequency of occurrence of features within any given data class which does not necessarily capture the most effective discriminant information within a data ensemble. For example, the FSN will receive the largest learning stimulus for low entropy features common to all patterns, regardless of class - such features contain little or no discriminant information. Thus, this method is unsatisfactory for data sets which have a large DC component.

This technique requires the user to adjust several parameters to optimise the performance of the system. To do this we must have some metric to gauge the performance of the system. This is just not possible for our particular application. Indeed, this is true of many Self Organising systems and it is often the case that one must have some idea of the inherent structure within the data to ensure that the net is behaving as it should. Note that this is not the case with the nodes presented in 3.5 since they do not attempt to cluster the data, merely to reduce its dimensionality. Ideally, for a practical system we would like its operation to be completely autonomous which is not possible where a large number of data dependent network parameters need to be set by the user.

6.2.2. Tamboratzis' SOLNN

Tamboratzis presents a SOLNN which responds to the frequency of occurrence of tuple features seen during training *[71]*. Like Kerins' FSN's, the sites store integers which are incremented or decremented during training.

In 3.3.3 we saw that learning in Self-Organising systems is a direct consequence of nodes which compete for a limited learning resource. To ensure that the resource constraint is satisfied, the process of learning data within the network requires that we must relinquish other, less important, data already learnt by that node. In Kerins' system this was achieved by incrementing the site addressed by the tuple feature and decrementing all other sites. In Tambouratzis' Network the constraint is satisfied by decrementing the site whose feature is most orthogonal to the one addressed. In this node the sites are not permitted to become negative and sites to be decremented are scanned, gradually decreasing the Hamming distance, until a non-zero site is encountered whereupon it is decremented.

These nodes preserve the relative distance relationships between the clusters in the input space (popularly referred to as a 'topological' relationship) through a process of localised lateral interaction between the nodes. This is analogous to the topological feature maps presented by Kohonen *[75][76]*. Each of the nodes lie on a 1-dimensional array and learning occurs in the neighbourhood of the node with the highest response to a given input pattern. As learning proceeds the size of the neighbourhood is gradually decreased so that the nodes settle into a stable solution in a fashion similar to simulated annealing in Boltzman Machines *[48]*.

This method has been most effective in clustering the very same character data set used throughout this thesis (see fig. 4.2, for example).

The size of the feature map array must be specified *a priori* and, since the network reflects the topological relationships within the data, it is often the case that, for a given data set, some nodes do not contribute during data classification. Some experimentation is required to optimise the size of this array so that the memory requirements of the system can be minimised. The precise configuration is data dependent.

Again, like Kerins' FSN's, this is a clustering network where learning is motivated by the frequency of occurrence of features rather than by the discriminant information conveyed by that feature. It too requires extensive experimentation in order to select the best system parameters for any given data set. Consequently, it is not particularly appropriate for our application.

6.2.3. Allinsons' logical Kohonen clone.

The popularity of Kohonens' topological feature maps has spawned Logical Neural Network implementations of this topology and associated learning paradigm. Allinson *[70][77]* presents a network, shown below in fig.6.1, in which each site within a discriminator contains an array analogous to a feature map .

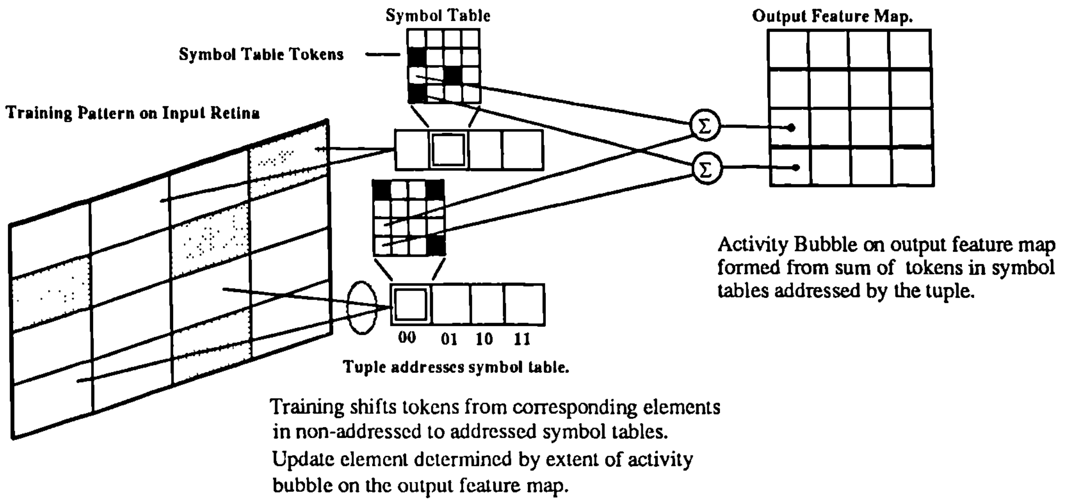

Symbol Table
Output Feature Map.
Symbol Table Tokens
Training Pattern on Input Retina
Σ
Σ
Activity Bubble on output feature map formed from sum of tokens in symbol tables addressed by the tuple.
00 01 10 11
Tuple addresses symbol table.
Training shifts tokens from corresponding elements in non-addressed to addressed symbol tables. Update element determined by extent of activity bubble on the output feature map.

Fig. 6.1 Basic architecture for Allinsons Self Organising Logical Neural Network.

A fixed number of single bit tokens are placed (at random initially) within elemental feature maps addressed by each tuple. These elemental feature maps are referred to as symbol tables. The activation level in each level of the output feature map is just the sum of the corresponding elements in the symbol tables addressed by each pattern. 'Activation bubbles' will tend to form in the output feature map, the highest of which is selected for update. The winning element and its neighbourhood are updated by filling empty sites in the update area of the addressed symbol tables with single tokens. A learning constraint keeps the number of tokens fixed and is imposed by shifting tokens from corresponding elements in other symbol tables across the tuple addresses. To prevent learning saturation due to DC components in the input ensemble, tokens are only shifted from symbol tables addressed by tuples with the same number of black and white pixels as the input.

This rather enigmatic learning law has been shown to cluster binary images quite effectively *[77]*. However, since each site within every tuple must contain a symbol table with the same dimensions as the output feature map, it does require large amounts of memory.

Tokens are shifted around during training and this requires an extensive search of the $[0,1]^n$ feature space during each data presentation - this means that training times are likely to be very long. Indeed, all of the method described here suffer from this drawback since the normalisation process required by the learning constraint involves a search of the $[0,1]^n$ space so that training times increase exponentially with tuple size.

6.2.4. A further logical implementation of a Kohonen feature map.

Another example of a Kohonen-type feature map implemented on a Logical Neural Network architecture is presented by Ntourntoufis *[72]*. In this model each node within the 2-dimensional feature map is represented by a discriminator output. The sites within the discriminator take on positive or negative real-values. Like the two previous models, the update occurs in the region of the node in the feature map with the highest response. The effect of the learning is spread out in the feature space as well by incrementing sites commensurately with the Hamming distance from the addressed site. The further the Hamming distance, the lower the learning stimulus. A cut-off point occurs beyond which the learning stimulus is zero. Beyond this, for a limited Hamming distance, sites are decremented by a small amount (this bears some resemblance to the site update strategy used in Kerins' FSNs and Tamboratzis' scheme).

As it stands, this learning rule makes no provision for normalisation. As a consequence, sites values can continue to increase unchecked which means that the system may not converge to a stable solution. This is quite a drawback.

6.2.5. On the formalism of learning paradigms for LNNs.

By and large it is true to say that the development of SOLNN learning paradigms (and to a lesser extent unsupervised learning in LNN's) has been motivated more by pragmatic considerations than through an analysis of the fundamentals governing the learning dynamics within such systems. This has come about because:

i. The WISARD system remains on of the few hardware realisable neural networks to date and certainly the only one which can be trained on large patterns (in excess of 0.25 million bits) at video frame rates at a modest cost. The impressive performance of the WISARD system has come about more through the application of sound engineering principles to tackle real world problems than by a slavish analysis of the learning dynamics. To an extent, the results achieved to date (especially with regard to the earlier work in this field) seem to validate the thrust of this approach.

ii. The highly non-linear transform produced by the tuple operator renders formal analysis extremely difficult. Indeed, this is also true of many of the sum-of-weights nodes and researchers in this area often limit analysis to simple linear nodes.

Nevertheless, the propensity of LNN researchers to steer away from the fundamental issues has tended to colour the nature of the work carried out to date. As a result many of the proposed architectures and associated learning laws have an *ad hoc* feel about them where the processes going on within the discriminator are not well understood. Consequently, it is difficult to asses those factors which control the efficacy of the learning mechanism such as convergence to optimal solutions, training data constraints and saturation problems. This dearth of theoretical foundation has dogged the development of robust learning strategies for LNN's.

In its basic form, the success of the WISARD lies with the simplicity of its 'one-shot' training methodology. It is not a sophisticated learning rule, by any means, but what it lacks in sophistication it makes up for in speed. Unfortunately, the crudeness of one-shot learning is not well suited to more complex learning techniques such as Self Organisation; here we are forced to opt for a more thorough approach to training where we cannot ignore the context of a pattern with respect to the complete data set.

The success of the n-tuple technique is a direct result of the expanded $[0,1]^n$ space. However, it is frequently true that the more complex training strategies require a search of this space (in the normalisation stage of the SOLNN's presented earlier, for instance). This can be rather time consuming. Thus, what was once a blessing is now a curse and the very attributes which make LNN's attractive for one type of learning system render them unfeasible for another. This begs the following question ' Is Self Organisation in LNN's really more effective than in their sum-of-weights type counterparts, both in terms of speed of training, memory requirements and optimality of solution?'. In this chapter we hope to go some way toward answering this question in the context of our application.

How suited are existing SOLNN architectures to our image coding scheme? We have seen that these architectures all act as data clustering systems rather than dimensionality reducers. It must be said that clustering is a limiting case of dimensionality reduction but the granularity of the clusters means that we will not get sufficient selectivity from the code (compare this with the classifier/dimensionality reducer argument presented in 5.4.3). In an attempt to confront these issues we aim to show how we can develop, from first principles, a Hebbian based learning paradigm for LNN's which is suitable for our coding scheme. This work is presented below.

6.3. Hebbian learning for Logical Neural Networks.

The role of a discriminator in an LNN based coding scheme is to partition the entire pattern space, represented by the spread of images within the database, so that we might encode each image in a manner which provides optimum selectivity. Hebbian Learning, which correlates the input and output activity of a node, produces a code which will convey the maximum amount of information across the database. For an image database whose contents form a Gaussian distribution in the pattern space this may be achieved by maximising the variance of the discriminator output. Let us take a closer look at how we might implement a Hebbian learning scheme within a Logical Neural Network.

In its basic form, Hebbian learning will accentuate learning when the output of the system is large and regulate it where it is small. In a simple LNN this can be implemented by updating the sites according to the magnitude of the discriminator output so that:

$$\Delta \mathrm{Site}_{addressed} = \beta\, y \qquad (38$$

Where:

β is a constant

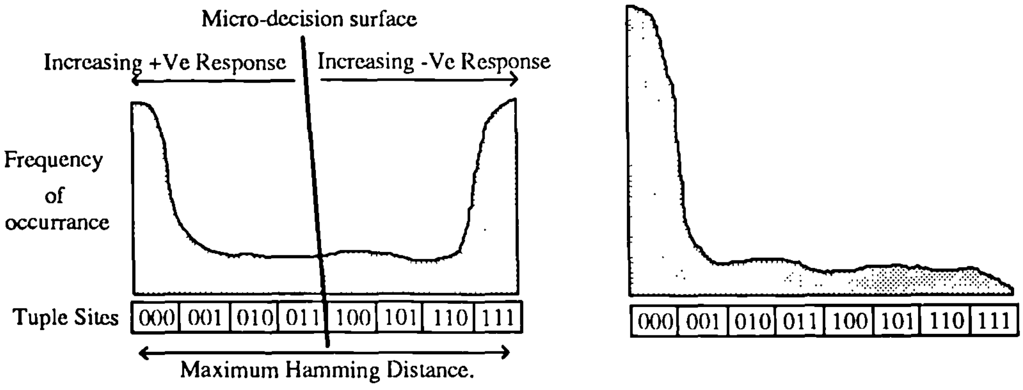
y is the discriminator response to a given pattern:

However, in common with the unconstrained Hebbian Learning law outlined in 3.3.2, the system will quickly saturate as the predominance of an output with a particular polarity induces a learning runaway where all of the sites begin to take on the same polarity. Added to this, we have seen that it is probable that some tuples provide almost no discriminant information at all and should not contribute to the learning process. How can we regulate learning in those tuples where the discriminant information is low?

6.3.1. Discriminant features in Self Organising Logical Neural Networks.

The decision surface learnt by a LNN during training may be regarded as the aggregate of the micro-decision surfaces learnt by each tuple within the discriminator. In order to develop a global decision surface, we must attempt to form good micro-decision surfaces within the functions of a single tuple. It is the job of the learning law to position this decision surface during training.

The most effective micro-decision surface is a partition which intersects frequently occurring but orthogonal features. This is shown below in fig.6.2. A learning law which partitions the data in this fashion will tend to maximise the variance of the discriminator output - this will improve the selectivity of the code. In an LNN, a feature may be regarded as a site within a RAM addressed by a tuple. The frequency that a particular site has been addressed during training gives a good indication as to where a partition might best be placed locally.



Frequently occurring equal and opposite tuples offers good prospect for decision surface.

A large DC component provides a poor prospect for decision surface.

Fig. 6.2 Contrasting good and bad micro decision surfaces within a tuple.

In Tambouratzis' model the decision surface is constructed by incrementing the site addressed by the tuple and decrementing the site furthest from it in the Hamming space.

It is not enough to partition the image space by simply attributing equal and opposite values to frequently occurring sites either side of this boundary as this gives no indication as to the co-occurrence of good partition features across the whole discriminator. Rather, learning should attempt to position the decision boundary to provide good local partitions with the constraint that they have a high co-occurrence with other local partitions across the discriminator. In Tambouratzis' model this is achieved by updating discriminators in the locality of the one with the highest response. It follows that a site should be updated (in the direction of the discriminator output) when:

i. The partition within the tuple is good. That is, both the site addressed and a site some Hamming distance from it have a high frequency of occurrence across the data set.

ii. The output of the discriminator is large, indicating a high co-occurrence with other partitions.

We can determine the potential efficacy of a local partition surface by applying a simple metric, θ(a,b), to each and every pair of sites, a and b, within a tuple prior to training. This is given in eqn. (39

$$\theta(a,b)=P_a.P_b.\left(\frac{H(a,b)}{\|P_a-P_b\|+\delta}\right)^2 \qquad (39$$

Where:

P_a	is the probability that site a is addressed. That is: P_a=(No. of times site 'a' is addressed)/(Total No. training patterns).
P_b	is the probability that site b is addressed.
H(a,b)	is the Hamming distance between sites a and b.
δ	is a small positive value to stop the function from going to infinity.

This function will give a high value when sites a and b are:

i. Addressed frequently.
ii. Addressed with approximately the same frequency
iii. Orthogonal in Hamming space.

Only those sites whose partition metric exceeds a threshold are considered for update. This is rather similar to the scheme proposed by Tattersall *et al [67]*. The learning runaway problem, discussed in Section 3.3.2, can be alleviated by updating the partition functions in pairs so that incrementing the addressed site infers that the other site in the partition pair is correspondingly decremented and *visa versa*. This learning law may now be encapsulated as:

i. Run through entire data set and record the frequency of occurrence of each site within the tuples.

ii. Calculate the partition metric of each and every pair of sites within the tuples. If the largest metric within a tuple exceeds a threshold then flag that pair as a partition pair eligible for update. Only one pair is selected per tuple.

iii. Seed each and every partition pair with a +1 and -1, selected randomly.

```
iv.   Present an image to the discriminator.
v.    If a tuple addresses one site of a partition pair then update them as
      {
            If Discriminator O/P < 0{
                  $\Delta Site_{partition\ addressed} = -1$
                  $\Delta Site_{partition\ not\ addressed} = +1$
            }
            If Discriminator O/P > 0{
                  $\Delta Site_{partition\ addressed} = +1$
                  $\Delta Site_{partition\ not\ addressed} = -1$
            }
      }
vi.   Repeat step v. for each and every image within the database.
vii.  Scan each partition pair within the discriminator.
      If partition site > 0{
            Clip site value to +1
      }
      If partition site < 0{
            Clip site value to -1
      }
viii. Repeat steps iv. to vii. until convergence.
```

Note that step vii. truncates the site values to prevent the magnitude of the site values from increasing as training proceeds.

It is instructive to compare the performance of this system with the sum-of-weights type node outlined in 3.4.4. The data set for this test, identical to that used in 4.2.1. (consisting of an 'A' font merged to a 'B' font over 25 images), was presented to the discriminator and the sites trained using the learning law described above. The 4-tuple discriminator converged after just 4 passes through the data set. The results are shown below in fig.6.3.

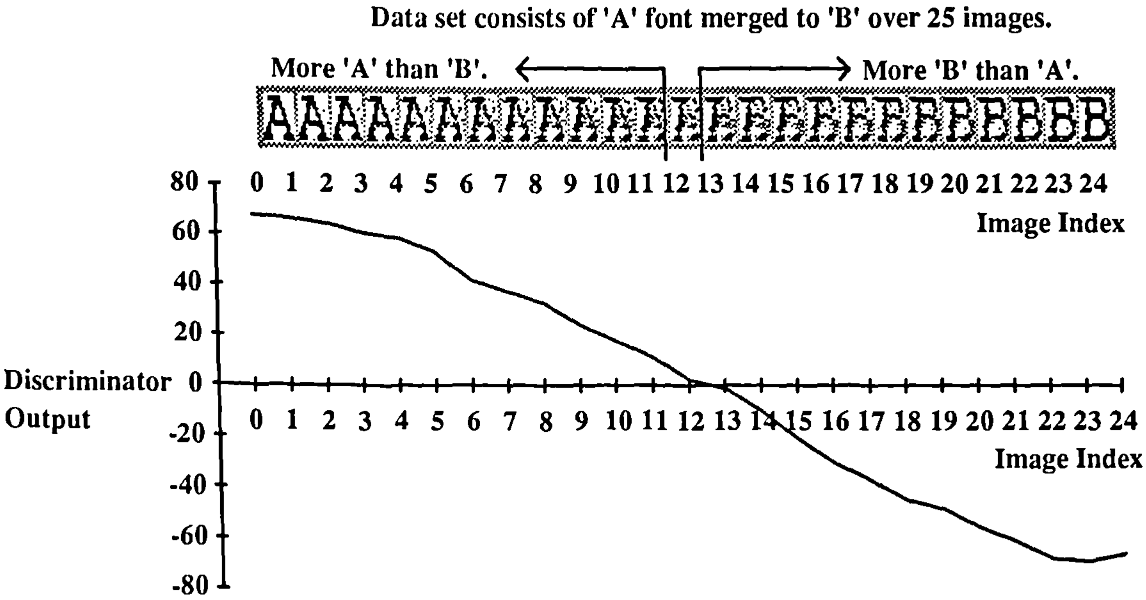


Fig. 6.3 Response of a single 4-tuple node trained using the learning law in 6.3.1

Although these results do compare favourably with those presented in 4.2.1., the learning law is rather inelegant and has an *ad hoc* feel about it. This is due partly to the pre-requisite that only partition sites earmarked during the initialisation phase are considered for update during the training phase. This sifting process relies only on local information and does not take into account the co-occurrence of other features within the discriminator. The learning law does correlate the co-occurrence of partition pairs during the update but some information is undoubtedly lost which will impair the performance of the code. In any event, selection of the partition metric threshold is unlikely to be a straightforward process. This is a distinct disadvantage for a practical system which is to be used with only a moderate level of expertise.

In an attempt to circumvent some of these difficulties we now turn our attention to an entirely novel learning law.

6.4. A New Self Organising Hebbian Learning Law from First Principles.

6.4.1. Developing the learning law.

A simple 1 tuple system to support the Hebbian learning is shown below in fig. 6.4. It consists of an array of real valued memory elements which are addressed by the binary image presented to the input retina. Location 0 to each memory is addressed by a 'white' pixel and location 1 by a 'black'. The sites can take on positive or negative real values and are initially filled, at random, with '-1's and '+1's.

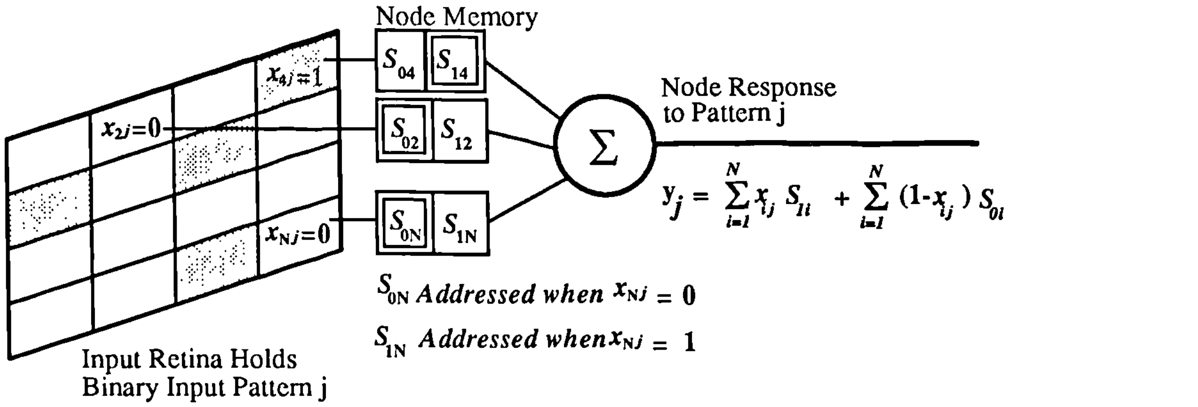

Node Memory
S_{04}
S_{14}
S_{02}
S_{12}
S_{0N}
S_{1N}
$x_{4j}=1$
$x_{2j}=0$
$x_{Nj}=0$
Σ
Node Response to Pattern j
$y_j = \sum_{i=1}^{N} x_{ij} S_{1i} + \sum_{i=1}^{N} (1-x_{ij}) S_{0i}$
S_{0N} Addressed when $x_{Nj} = 0$
S_{1N} Addressed when $x_{Nj} = 1$
Input Retina Holds Binary Input Pattern j

Fig. 6.4 Basic architecture for the Self Organising Logical Neural Network.

The thrust behind Hebbian learning is to update the addressed site commensurately with the strength of the node response. This can be achieved by adding the node response to a particular image to the sites addressed by that image. For the architecture shown in fig. 6.4 the learning law becomes.

$$\Delta S_{1i} = \gamma . y_j x_{ij} \qquad \text{(40}$$

$$\Delta S_{0i} = \gamma . y_j (1 - x_{ij}) \qquad \text{(41}$$

The output of the node is given by:

$$y_j = \sum_{i=1}^{N} S_{1i} x_{ij} + \sum_{i=1}^{N} S_{0i} (1 - x_{ij}) \qquad \text{(42}$$

Where:

ΔS_{1i} is the update on the site of the site addressed when the i_{th} pixel in j_{th} image is 1 (i.e. black).

ΔS_{0i} is the update on the site of the site addressed when the i_{th} pixel in j_{th} image is 0 (i.e. white).

y_j is the node response to the j_{th} image.

x_{ij} is the value of the i_{th} pixel in the j_{th} image
$x_{ij}=1$ for a black pixel.
$x_{ij}=0$ for a white pixel.

N is the dimensionality of the images within the database.

γ is a constant to limit the dynamic range of the sites.

However, such a learning law will tend to accentuate the effect of those areas within the image which have a large DC component and contain no discriminant information. A predominance of a pixel 'colour' across all images will cause the output to saturate exponentially rapidly. Such a node will give the same response to each and every image within the database. This can be circumvented by forcing the mean of the node response to zero so that the total update for all of the sites addressed by a tuple is zero. Thus, a site which is addressed by every image within the data base will contain zero after one pass of the data set and will not contribute to the output of the node. This update strategy will filter out unwanted DC components from within the data set *[78][79].*

After one pass through the data set the new site values become:

$$\hat{S}_{1i} = \gamma \sum_{j=1}^{P} (y_j - \bar{y}) x_{ij} \qquad (43$$

$$\hat{S}_{0i} = \gamma \sum_{j=1}^{P} (y_j - \bar{y})(1 - x_{ij}) \qquad (44$$

Where

$\bar{y}$ is the mean response of the discriminator to the entire data set. That is:

$$\bar{y} = \frac{1}{P} \sum_{j=1}^{P} y_j \qquad (45$$

P is the number of patterns in the training set.

The complete training cycle may be summarised as:

i. Clear down the discriminator by setting each memory location to 0.
ii. Seed the node by writing a 1 to those sites addressed by an image chosen arbitrarily form the database.
iii. Present each pattern to the discriminator and record the response of the discriminator.
iv. Clear down the discriminator as in stage i.
v. Re-present each pattern to the node and add the normalised, mean-corrected response to the site addressed by that pattern.
vi. Repeat stages ii. to vi. until convergence.

6.4.2. Analysing the learning law for a 1-tuple system.

A fraction of the mean-corrected output of each node is added to the addressed site on each training presentation. Thus, after one pass through the complete data set, the sum of the sites values addressed by each tuple is zero, i.e. for a 1-tuple system:

$$\hat{S}_{1i} + \hat{S}_{0i} = 0 \qquad (46$$

Then from equations (42 and (46:

$$y_j = \sum_{i=1}^{N} S_{1i}(2x_{ij} - 1) \qquad (47$$

Combining equations (45 and (47

$$\bar{y} = \frac{1}{P}\sum_{i=1}^{N} S_{1i} \sum_{j=1}^{P} (2x_{ij} - 1) \qquad (48$$

So that:

$$y_j - \bar{y} = 2\sum_{i=1}^{N} S_{1i}(x_{ij} - m_i) \qquad (49$$

Where:

m_i is the mean number of black pixels in the i_{th} element in the retina across the complete data set. That is:

$$m_i = \frac{1}{P}\sum_{j=1}^{P} x_{ij} \qquad (50$$

Substituting equation (49 into (42 gives the new value of the site addressed by the black pixels after one complete pass through the data set as:

$$\hat{S}_{1k} = 2\gamma \sum_{i=1}^{N} S_{1i} \sum_{j=1}^{P} x_{kj}(x_{ij} - m_i) \qquad (51$$

In matrix notation the state equation for the learning rule becomes:

$$\hat{S}_{1k} = 2\gamma[S_{11} \;\; .. \;\; S_{1N}][x_{k1} \;\; .. \;\; x_{kP}]\begin{bmatrix} x_{11} - m_1 & .. & x_{1P} - m_1 \\ . & & . \\ x_{N1} - m_N & .. & x_{NP} - m_N \end{bmatrix}^T \qquad (52$$

Since x_{ij} is binary valued this is equivalent to:

$$\hat{S}_{1k} = 2\gamma[S_{11} \;\; .. \;\; S_{1N}][x_{k1} - m_k \;\; .. \;\; x_{kP} - m_k]\begin{bmatrix} x_{11} - m_1 & .. & x_{1P} - m_1 \\ . & & . \\ x_{N1} - m_N & .. & x_{NP} - m_N \end{bmatrix}^T \qquad (53$$

The vector state equation for this learning law can be written as:

$$\hat{\mathbf{S}}_1 = 2\gamma \mathbf{S}_1 \mathbf{Q} \qquad (54$$

Where:

$\hat{\mathbf{S}}_1$ is the new site vector $\begin{bmatrix} \hat{S}_{11} & \cdot\cdot & \hat{S}_{1N} \end{bmatrix}$ after a complete training cycle.

$\mathbf{S}_1$ is the site vector $\begin{bmatrix} S_{11} & \cdot\cdot & S_{1N} \end{bmatrix}$ prior to the training cycle.

$\mathbf{Q}$ is the auto correlation matrix of the data set. That is:

$$\mathbf{Q} = \begin{bmatrix} x_{11}-m_1 & \cdot\cdot & x_{1P}-m_1 \\ \cdot & & \cdot \\ x_{N1}-m_N & \cdot\cdot & x_{NP}-m_N \end{bmatrix} \begin{bmatrix} x_{11}-m_1 & \cdot\cdot & x_{1P}-m_1 \\ \cdot & & \cdot \\ x_{N1}-m_N & \cdot\cdot & x_{NP}-m_N \end{bmatrix}^T \qquad (55$$

Since the update of the sites addressed by the white pixels is directly equivalent to the update of those sites addressed by the black pixels we can write:

$$\hat{\mathbf{S}}_0 = 2\gamma \mathbf{S}_0 \mathbf{Q} \qquad (56$$

or

$$\hat{\mathbf{S}}_0 = -2\gamma \mathbf{S}_1 \mathbf{Q} \qquad (57$$

Empirical tests have shown that setting γ as:

$$\gamma = \frac{2}{\sum_{j=1}^{P} \| y_j \|} \qquad (58$$

will limit the dynamic range of the site values so that $-1 \le S_0 \le 1$ and $-1 \le S_1 \le 1$.

Learning rules with state equations of the type shown in equation *(56* have been thoroughly investigated and proven to converge to the eigenvectors of the auto correlation matrix of the data set *[41][53][56]*. This update rule, based on Hebbian type learning, is directly equivalent to PCA and has some rather interesting features which render it a more attractive prospect for our coding scheme than both classical and existing neural network implementations of this transformation *[78][79]*.

i. Unlike classical PCA our technique does not require the direct computation of the auto correlation matrix nor its eigenvectors which is impractical for high dimensional data sets.

ii. Unlike existing sum-of weights type neural network implementations of PCA, no implicit approximations are made in the derivation of the state equation for the leaning rule. The problems associated with this approximation have been discussed in 3.4.2. The upshot of this is that the learning rule does not require a learning rate to be set which, in turn, leads to faster and more accurate convergence.

iii. Most neural network paradigms are sensitive to the order of presentation of the data set and are prone to settle into false minima, especially if the learning rate is set too high (see 4.2.4). Because this learning rule does not implement the system updates until all of the data has been presented, it does not suffer this drawback and consequently will not converge to localised solutions. This is, in fact, a reinterpretation of axiom ii above.

It is evident that equations *(54* and *(56* are effectively equivalent to each other since the update of the S_1 sites is inexorably bound to the update of S_0. One could argue, and with good reason, that the memory requirements of this system could be halved simply by considering update on the S_1 sites only. Whilst this is undoubtedly true for the simple 1-tuple system, the notion that the white and black elements address distinct memory locations allows the learning law to be applied to systems with larger tuple sizes.

6.4.3. Extending the novel learning law to larger tuple sizes.

The learning law developed in 6.4.2 can be readily applied to systems with larger tuple sizes, and although the non-linearities induced by these higher order systems render the proof of convergence somewhat incomplete, the system dynamics are, ostensibly, still the same. Bearing in mind that these non-linearities will degrade the conformality of the transformation produced by the node (see 5.4.4), one might ask why we should even consider employing larger tuple sizes. This point has been dealt with in some depth in 5.6.2 which concluded that the greatly expanded feature space (often referred to as $[0,1]^n$ space) afforded by larger tuple sizes can assist in orthogonalising learning in subsequent nodes for multi-node systems. As the tuple size increases the degradation of the conformality of the mapping will begin to offset the advantages afforded the expanded feature space.

All existing learning laws using LNN's require some scanning of this feature space in an attempt to normalise the response of the node. As the tuple size grows then the time taken to scan this space increases exponentially. This is rather unfortunate as one of the prime reasons for using Logical Neurons in the first place is their potential for rapid training. However, the learning law developed here does not suffer this drawback since a site update does not require information from any of the other sites within the tuple.

Normalisation is achieved automatically by virtue of the fact that the mean of the total update within any tuple is forced to zero - this is implicit in the learning law. Since the number of tuples decreases with tuple size and, recalling that the training time is dependent only upon the number of tuples within a discriminator, it is apparent that, for this learning algorithm, training times actually decrease as the tuple sizes gets larger.

The learning law for any tuple size is exactly the same as the 1-tuple outlined in 6.4.2 The effect of larger tuple sizes on the nodal transformation can be seen by repeating the test described in 4.2.3 using the learning law developed here with a range of tuple sizes. The results are shown below in fig 6.5. In these experiments the tuples are mapped to maximise the mean Euclidean distance between the sites assigned to each tuple. This mapping strategy produces the best results - the reason for this is discussed in 5.6.3.

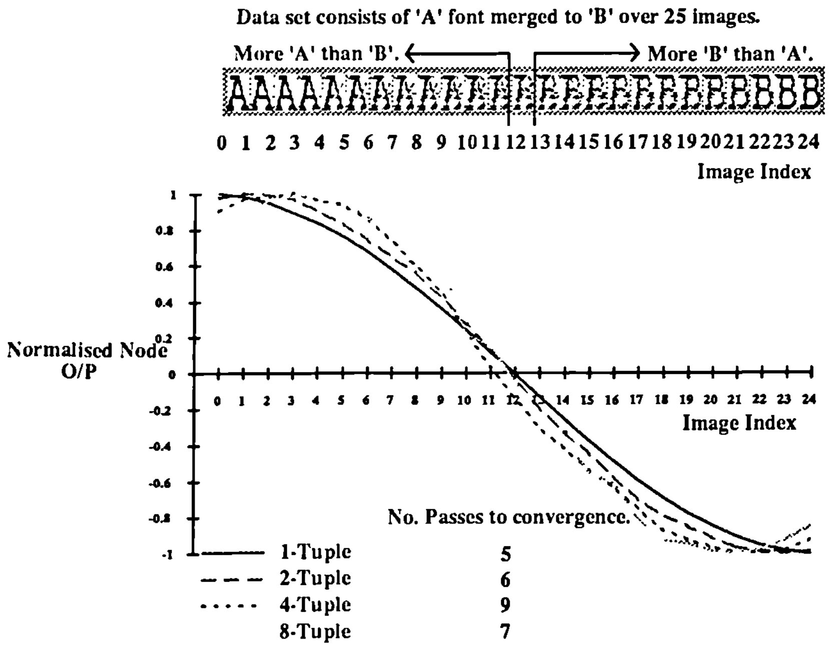


Fig. 6.5 Variation of node response with tuple size for a single node Logical Neuron.

These results shown that as the tuple size increases, the node begins to behave more like a classifier than a dimensionality reducer. It is evident that the relative distance relationships between images in the original pattern spaces are not preserved as the tuple size increases. This effect is most pronounced where the output of the discriminator is large.

6.4.4. Extending the learning law to multi-node systems.

Successive nodes in a multi-node Self Organising Neural Network should learn orthogonal features within the data set. In the sum-of-weights network, outlined in 3.4.6, this was achieved by subtracting a fraction from the input dependent upon the magnitude of the output and the weight on the input line on 'earlier' nodes. Sanger*[41]* has shown that this strategy produces a transform analogous to Gram-Schmitt orthogonalisation.

Unfortunately, this technique cannot readily be applied to logical nodes as their inputs must be binary to address the sites within each tuple - we cannot remove fractions of the input for presentation to successive nodes. One rather crude, but nevertheless quite effective, expedient is to employ a 'winner-take-all' type learning scheme where only the node with the largest output is considered for update. In this scheme nodes compete for learning stimulii and orthogonalatilty is achieved by virtue of the fact that a single node will learn a feature type specific to that node and no other. The automatic normalisation implicit in the learning law prevents the range of feature types learnt by that node from becoming excessively broad (where the same node responds maximally to each and every input image). Apart from this restriction, the learning law is identical to that presented in 6.4.2, except that each node must be seeded differently to induce competitive learning between the nodes. Note also that learning (and the calculation of the mean output of the node etc) is only engaged for 'winning' images.

This winner-take-all type learning paradigm has the advantage that training times are not directly dependent upon the number of nodes used since only one node participates in the learning process. This is not the case with the learning law outlined in 3.4.6.

An image coding scheme using the SOLNN and accompanying winner-take-all learning law developed here was tested on the data set presented in fig. 4.2. Note that the tuples are mapped to maximise the mean Euclidean distance between the sites assigned to each tuple. The results are shown below in fig.6.6.

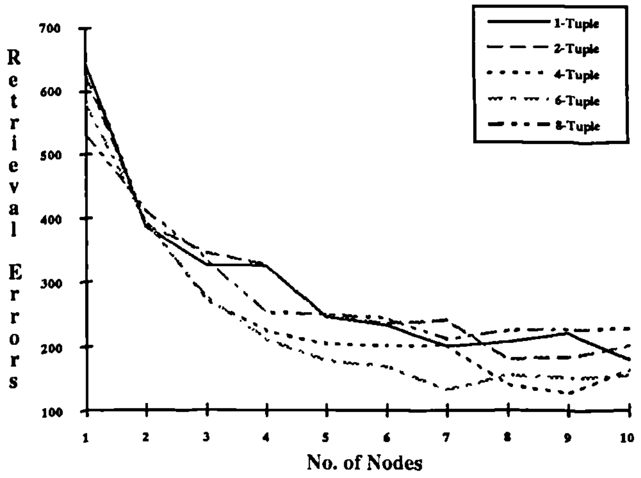


Fig. 6.6 Variation of code performance with number of nodes for Database shown in fig. 4.2

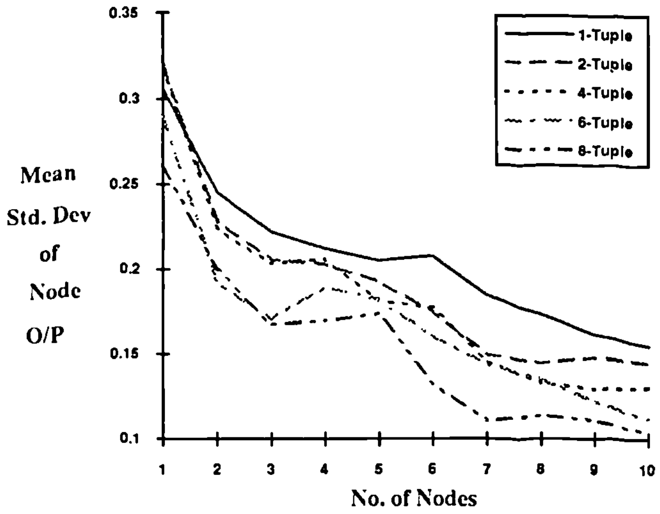


Fig. 6.7 Effect number of number nodes on the mean standard deviation of node responses for database shown in fig. 4.2

The convergence times for a broad range of system parameters using this data set is shown below in fig. 6.8. Convergence is deemed to occur when the RMS error of discriminator outputs for respective input patterns on successive learning cycles is less than 0.1%.

Tuple Size	No. of Nodes									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	33	17	15	13	13	13	12	16	17	12
2	31	18	17	14	17	18	29	23	20	20
4	24	17	16	19	13	12	12	15	20	11
6	65	20	14	13	17	15	21	22	21	21
8	30	21	24	17	17	24	22	20	20	20

Fig. 6.8 Variation of Convergence times (in data set cycles) with system parameters.

These results show that larger tuple sizes can improve upon the performance of the winner-take-all implementation of the LNN learning law. Unfortunately however, since there is no correlation between the data in figs. 6.6 and 6.7, it is difficult to select the optimum tuple size for any given data set using the standard deviation of the node response. Thus we are blighted by a problem which affects nearly all n-tuple learning systems, namely: 'How can we match the tuple size to the nature of the problem ?' There appears to be no easy solution.

Fig. 6.8 confirms that convergence times (in number of cycles of the training data) are, in general, dependent neither on the number of nodes in the system nor the tuple size. Again, it is difficult to highlight any relationship between these parameters which might help us to optimise the system performance.

6.4.5. Effect of database diversity on the performance of a winner-take-all implementation of the learning law.

It is useful to observe how the performance of the system developed in 6.4.4 changes as the contents of the database become more diverse. To this end, the experiment presented in 6.4.4 was repeated with the number of classes of characters increased to 24. The relationship between the number of nodes and system performance and the standard deviation of the node response is shown in fig. 6.9 and 6.10 respectively.

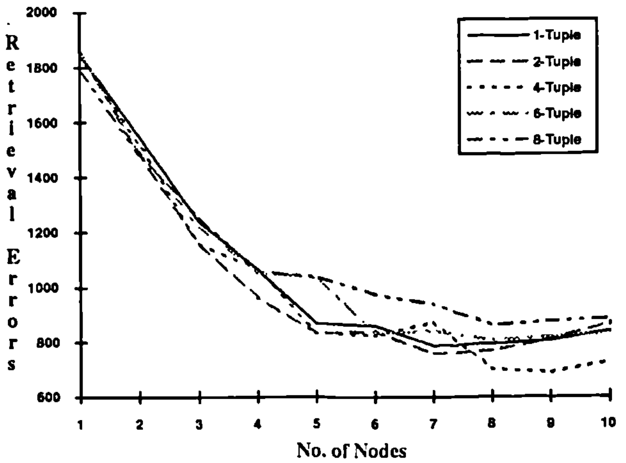


Fig. 6.9 Variation of code performance with number of nodes. Database consists of 10 examples each of 24 classes of characters.

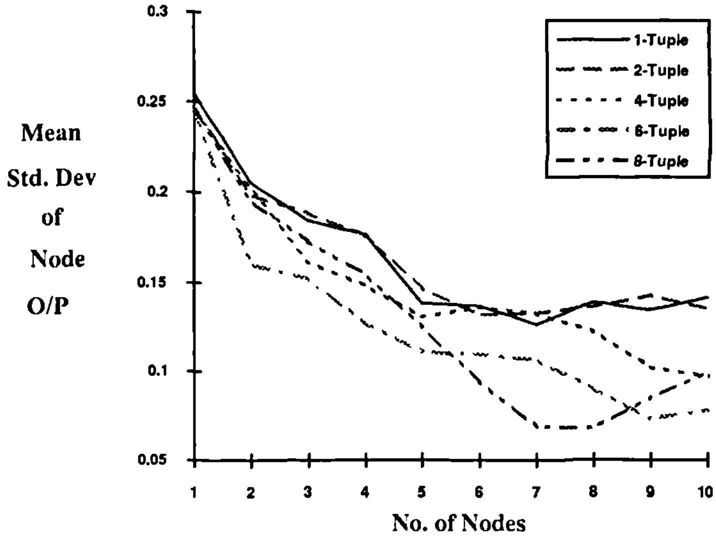


Fig. 6.10 Effect number of number nodes on the mean standard deviation of responses. Database consists of 10 examples each of 24 classes of characters.

Convergence times for this data set are given below in fig. 6.11.

Tuple Size	No.	of	Nodes							
	1	2	3	4	5	6	7	8	9	10
1	28	21	17	16	39	35	34	33	37	40
2	45	21	30	27	29	37	28	29	31	28
4	192	26	25	31	56	20	35	53	21	29
6	62	42	30	61	30	46	56	82	31	22
8	31	41	29	25	34	35	35	53	46	56

Fig. 6.11 Variation of convergence times (in data set cycles) with system parameters. Database consists of 10 examples each of 24 classes of characters.

6.4.6. What tuple mapping gives the best performance for the learning law ?

In Section 5.6 of this thesis we saw how different tuple mappings onto the input retina affected the performance of discriminator based LNNs. We concluded that the best performance could be achieved by de-correlating local image artefacts. This can be done rather simply by mapping the tuples to maximise the mean Euclidean distance between sites monitored by any one tuple. It is worth re-confirming this for the learning law developed here. Fig. 6.12 does indeed show that this mapping strategy produces the best results (note that a 6-tuple is the optimum tuple size for this data set).

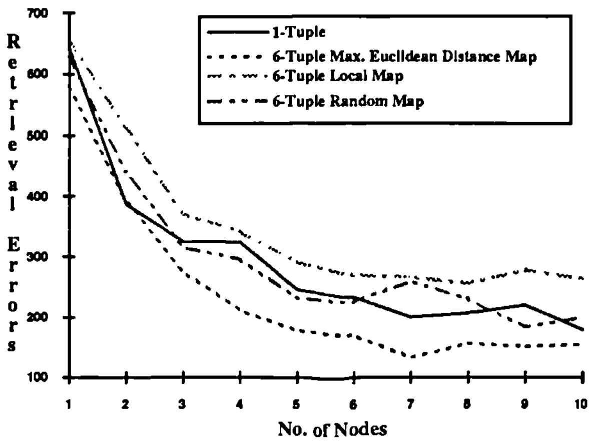


Fig. 6.12 Comparing the performance of different mappings for a 6-tuple system.

6.5. Applying the Learning Rule to a Sum-of-Weights Node.

The essence of our learning rule is pleasantly simple: during training we add a fraction of the mean corrected node output to the site addressed by that tuple. The mean of the node output is subtracted from the output prior to this update so that, after one training cycle, the sum across the 2^n sites addressed by an n-tuple is zero. The upshot of this for a 1-tuple system is that the contents of the sites addressed by 'black' and 'white' pixels are equal and opposite. One could argue that, because the value of the 'white' sites is inexorably bound to the value of the 'black' sites, only one of these sites is required. In this case the polarity of the of the site could be re-created by multiplying a single element memory by '+1' if the pixel is 'black' and '-1' if the pixel is 'white'. Such a node is directly equivalent to a standard sum-of-weights node - this is shown below in fig.6.13.

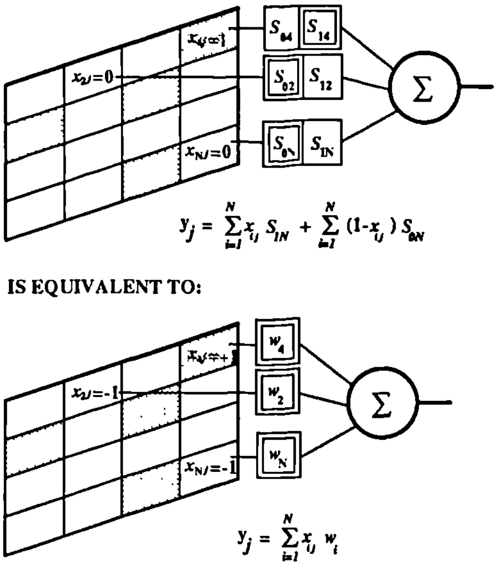


Fig. 6.13 A 1-tuple system is equivalent to a sum-of-weights node for our learning law.

For a single node system the learning law becomes:

$$w_{i,t+1} = w_{i,t} + x_{ij}\left(\frac{y_j - \bar{y}}{\kappa}\right) \tag{59}$$

Where:

$w_{i,t}$ is the value of the site before the update.

$w_{i,t+1}$ is the value of the site after the update.

x_{ij} is the value of the i_{th} pixel in the jth image.

y_j is the response of the j_{th} image obtained from stage ii.

$\bar{y}$ is the mean value of the node responses from stage ii. That is:

$$\bar{y} = \frac{1}{P}\sum_{j=1}^{P} y_j \tag{60}$$

P is the number of images within the database.

κ is a regulating term obtained from the node responses.

$$\kappa = \frac{\sum_{j=1}^{P}\|y_j\|}{2} \tag{61}$$

A comparison between the weights developed using this learning law for a 1-tuple system and Sangers' learning law, given in 3.4, reveals that both methods converge to the same solution with the exception that the weights are not normalised in the same way. In Sangers node, the sum of the square of the weights is equal to 1 whilst in the novel learning law $-1 \leq (Site \quad value) \leq 1$.

6.5.1. Why is a sum-of-weights implementation preferable to a 1-tuple system ?

The advantages of a single node sum-of-weights node over a 1-tuple LNN system are two-fold: firstly, the memory requirements are halved and, secondly, we are no longer restricted to binary images. This latter point is quite significant as, typically, a large amount of image information is lost in converting a grey-level image to a binary one.

In the sum of weights implementation of the learning rule presented in 3.5, orthogonal learning for multi-node systems achieved by removing a portion of the input to successive nodes according to the magnitude of the node response in 'earlier' nodes in the system. This technique is described in more detail in 3.5.2. This orthogonalisation technique is very effective but cannot be applied to an LNN scheme because the inputs on the tuples must be binary to form meaningful addresses to the sites. In an attempt to circumvent this problem we resorted to a 'winner-take-all' type learning system which is comparatively crude. However, since the inputs to the nodes on our revised sum-of-weights type node can take on any value between -1 and +1 we have the potential to develop a more sophisticated orthogonal learning scheme. This avenue is worthy of further investigation.

6.5.2. Orthogonal learning in a sum-of-weights implementation.

For Sangers' neural network architecture *[41]*, presented in 3.5.2, the input to successive nodes is regulated both by the magnitude of the node output and the value of a particular weight in earlier nodes. In this training scheme the weight vector lies on a valid trajectory toward a stable solution at each presentation of data. Such an update scheme cannot be supported by our node however because the weights only take on meaningful values at the end of a training cycle. Prior to this, the weight is not representative of the transformation learnt by the discriminator and cannot be used to calculate the orthogonal input to successive nodes. Essentially, we can only calculate the inputs to nodes further down the 'learning chain' if we have access to the weight vectors of nodes further up the chain at the end of the previous training cycles. There are two ways round this problem:

i. Consider a system where node 1 is the first node in the learning chain (learning the first principal component). We begin by presenting a complete training cycle to all the nodes but adapting the weights on odd numbered nodes. During this time learning is frozen on even nodes and the static weights used to calculate the orthogonal inputs to odd nodes. At the end of the training cycle the weights are representative of the transform learnt by these nodes and may be used to calculate the orthogonal inputs to the even nodes. The orthogonalising input is given below in equation (62. Learning is then suspended on odd node and the training process repeated for even nodes. Thus learning must be suspend on every alternate node at each pass through the data set so that the weight vector on the node preceding the one undergoing learning is fixed to calculate the orthogonalised input on the next node.

ii. We could buffer the value of the weight vector from each node (barring the first which does not receive an orthogonalised input) from the last training cycle and use this to calculate the orthogonalised input, given below in equation (62.

Both methods converge to the same solution. Method ii has twice the memory requirements of method i but converges twice as fast. The most appropriate method can be matched to the users requirements; in all the following simulations method ii has been used.

The orthogonalised input is given below:

$$x_{ijk+1} = x_{ijk} - w_{ik}\left(\frac{y_{jk} - \bar{y}_k}{\varsigma_k}\right)$$ (62

Where:

x_{ijk+1} is the value of the i_{th} pixel from the j_{th} image to the $k{+}1_{\text{th}}$ node in the system.

x_{ijk} is the value of the i_{th} pixel from the j_{th} image to the k_{th} node.

w_{ik} is the value of the i_{th} weight on the k_{th} node.

y_{jk} is the response of the k_{th} node to the j_{th} image.

$\bar{y}_k$ is the mean response of the k_{th} node.

That is:

$$\bar{y}_k = \frac{1}{P}\sum_{j=1}^{P} y_{jk}$$ (63

$$\varsigma_k = \sum_{i=1}^{N} {w_{ik}}^2$$ (64

N is the number of pixels in each image. The images must be normalised so that N is constant across the whole data set.

The architecture to support this learning rule is shown in fig 3.9.

6.6. Some Practical Results for the New Learning Law.

The tests outlined in 4.2.3 are repeated here for the learning law developed in 6.5.2. Comparing these results with those achieved by Sangers' learning law, given in figs. 4.10 and 4.11, show that the performance of the two schemes is remarkably similar. However, the learning law presented in this chapter does not require a learning rate to be set and this is a distinct advantage for a practical system.

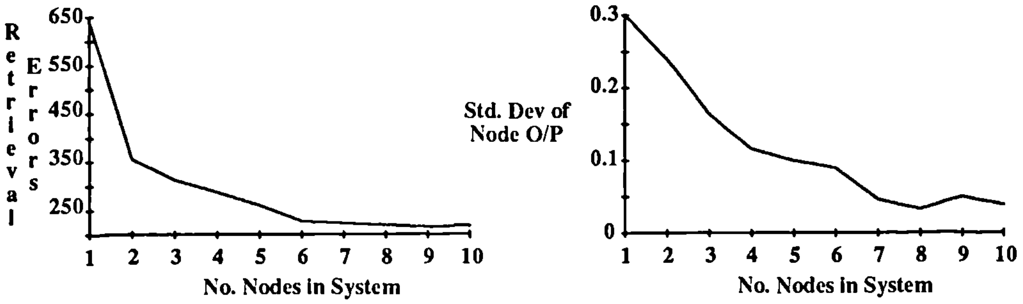


Fig. 6.14 Variation of code performance and standard deviation of node output with number of nodes for a sum-of-weights implementation of the new learning law

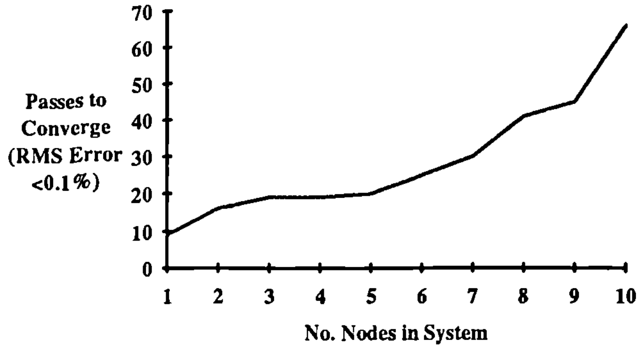


Fig. 6.15 Variation of convergence times with number of nodes for a sum-of-weights implementation of the new learning law

6.6.1. When should we re-train the neural network?

The great strength of our coding scheme lies in its ability to adapt to the changing nature of the database. Each of the images is coded with respect to a set of features learnt by the network. These features characterise the distribution of images within the database. If the contents should change, then the transform learnt by the network is likely to be sub-optimal and the neural network will have to be re-trained. Since re-training is, typically, a rather lengthy process, it is useful to investigate how the performance of the retrieval mechanism is effected where images are added to the database without re-training.

In 4.4.3 we argued that if the additional images are similar to the class of images already contained within the database, then the need for re-training is not particularly pressing. If, however, these images differ significantly then the degradation in performance is marked. In following experiment, a 6-node neural network was trained on the images shown in fig. 4.2 using the learning law outlined in 6.5.2. The data set consisted of 10 classes of 10 machine printed fonts. The object of this experiment is to observe the effect of adding images to an existing database. The following image sets were added (in sets of 20 at a time) to the database and the retrieval performance measured with and without training:

- Similar image types - a further 2 examples each of the 10 classes of fonts already present within the database. (i.e. add 2 different A's, 2 B's.....2 K's, measure the performance and repeat).

- Different image types - a further 2 examples each of 10 classes of fonts not found within the database (i.e. add 2 each L's, 2 M's....etc., measure the performance and repeat).

The results are presented below in fig. 6.16. In this test relative errors are defined as $\frac{No.\ of\ Errors}{Max.\ possible\ Errors}$.

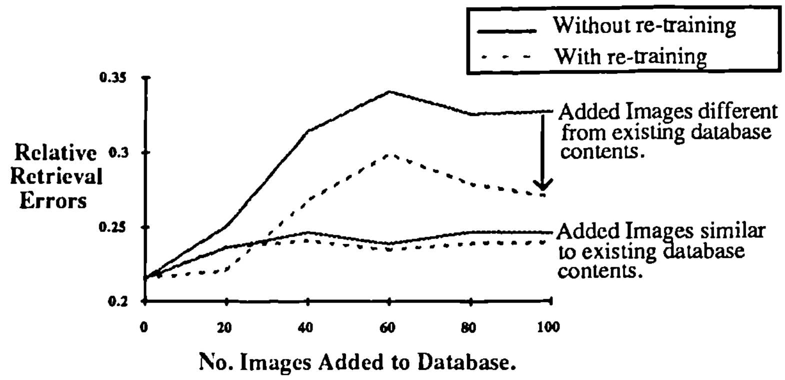


Fig. 6.16 The effect of re-training on an expanding database.

These results confirm the conclusions drawn in 4.4.3 and 4.4.4 of this thesis.

6.7. Comparing the Performance of Four Learning Systems Discussed in this Thesis.

In this section we present an objective comparison between the following learning laws:

i. Sangers Learning Law presented in Section 3.4.6 of this thesis (see Section 4.3 for practical tests).

ii. A 6-tuple LNN implementation of the Hebbian learning law presented in Section 6.2.3.

iii. A sum-of-weights implementation of the Hebbian learning law presented in Section 6.5.2.

iv. A 6-tuple LNN implementation where the training exemplars are selected from an exhaustive search of the database images. This training strategy is outlined in Section 5.8.

Tests were repeated for two data sets:

a. The data base shown in fig. 4.2, consisting of 10 examples each of 10 characters.

b. A data base of 20 examples each of 20 character classes ('A' to 'V' not including 'O' or 'I').

The results for data set a. are shown in figs.6.17, 6.18 and 6.19. Those for set b. are shown in figs.6.20 and 6.21 respectively. Note that the training times for the strategy outlined in 5.8 increase exponentially with the database size and linearly with the number of nodes and are significantly longer than the other 3 methods. This is shown for data set a. in fig 6.19.

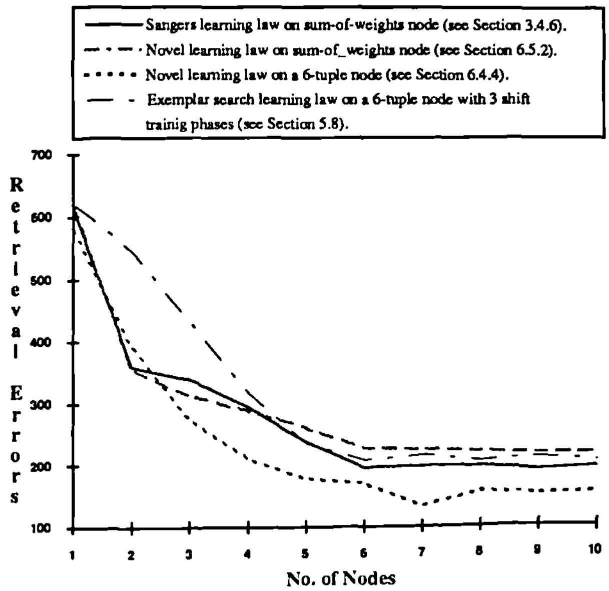


Fig. 6.17 Comparing the performance of 4 learning laws for data set a.

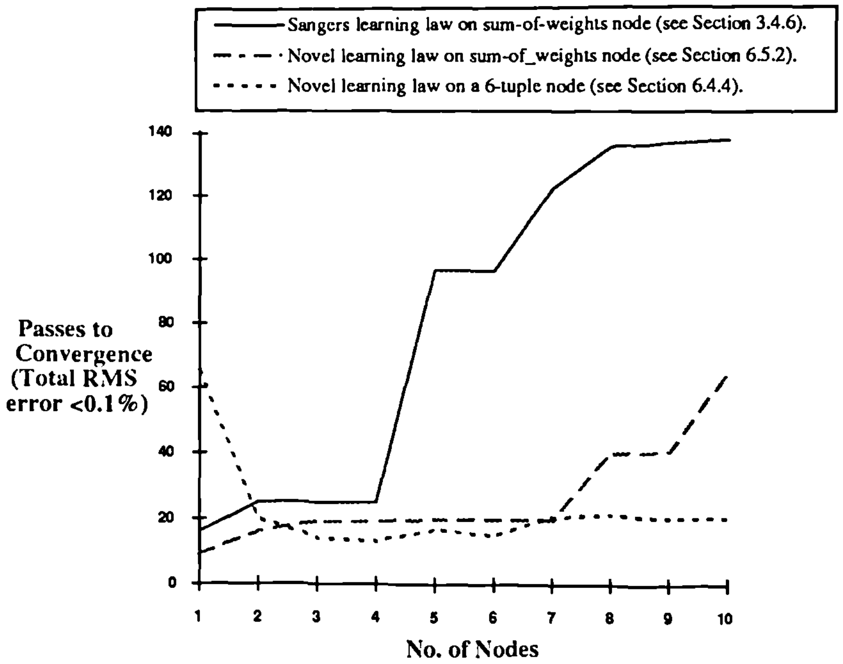


Fig. 6.18 Comparing convergence times for 3 learning laws using data set a.

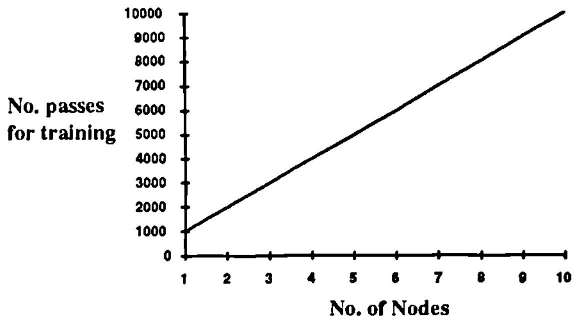


Fig. 6.19 Training times for the exemplar search training algorithm using data set a. (see Section 4.8)

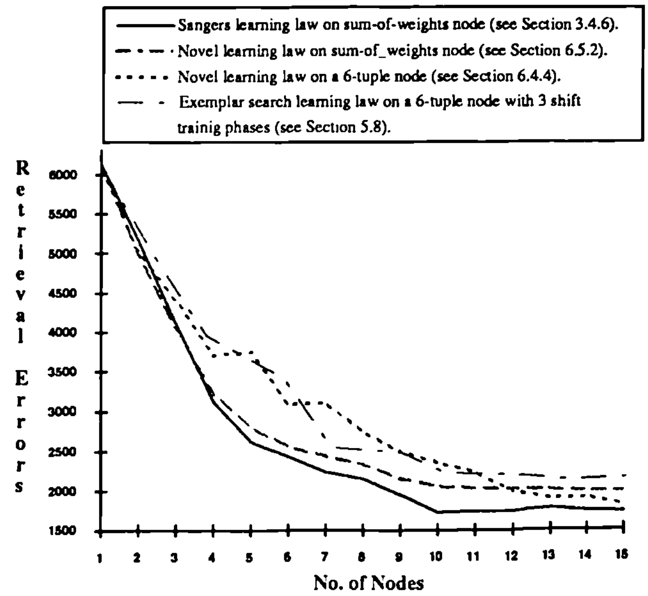


Fig. 6.20 Comparing the performance of 3 learning laws for data set b.

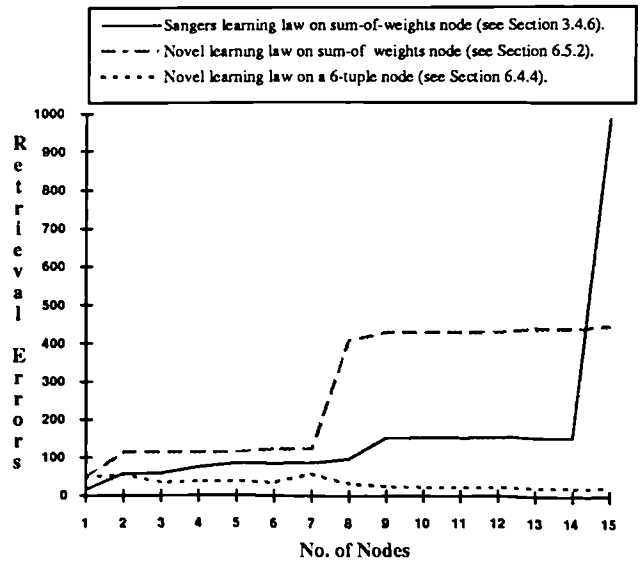


Fig. 6.21 Comparing convergence times for 3 learning laws for data set b.

6.7.1. Comparing our coding scheme with a mask matching system.

In essence, our coding scheme attempts to represent the images in a more manageable domain than the original to support rapid fuzzy matching across a very large database. Searching this new domain is undoubtedly quicker, but is it really more effective than comparing raw images using Hamming distance, for example?

The NNT is just an information filter which sifts out redundant data from an ensemble. The important issue here is: Does the information that the NNT filter out actually impair or improve the performance of the matching mechanism? In the following test we repeated the experiment outlined in 6.7 using the data set a. (shown in fig. 4.2) but matched the images through Hamming distance on the 'raw' images rather than using the codes. The best performance figures for this, and the previous tests is shown below in fig.6.22 (note that a floating point number is taken to be 4 bytes long and that each element of the code has been quantised to 8 bits).

The results show that a coding scheme using the novel learning law on 6 x 6-tuple nodes will outperform all other techniques presented here. The price that we pay for this is the increased memory requirements for the system. However, training times are significantly less than for Sangers' learning law (and, to a lesser extent, the other coding schemes) and this is a distinct advantage for a practical system. The mask matching scheme does not require any training but each comparison takes 8 times longer than all of the coding schemes. Thus, whilst the training is inconvenient, the overhead is only incurred intermittently whereas, for the mask matching scheme, the extended searching times are introduced each time a query is posed to the system. This is not desirable.

Matching/Coding Method	Retrieval errors	Memory for code (bits)	Comparisons for each match (bits)	Training times (passes of data set)	Memory for training (bytes)
Mask match - Hamming distance.	217	0	384	0	0
Code match - Sangers learning law on 6 sum-of-weights nodes	197	48	48	97	9216
Code match - Novel learning law on 6 sum-of-weights nodes.	221	48	48	20	9216
Code match - Novel learning law on 6 x 6-tuple Logic nodes.	178	48	48	17	98304
Code match - Exemplar search on 6 x 6-tuple Logic nodes.	204	48	48	6000	3072

Fig.6.22. Comparing mask and code matching strategies for an image retrieval system.

It is also important to note that the Hamming distance matching strategy can only be performed on binary images. The same holds true of the Exemplar search and LNN implementation of the novel learning law. This is something of a drawback as important information will be lost during thresholding which is likely to impair the performance of the code.

6.8. Is a 1-tuple Coding Scheme Really Performing Mask Matching ?

WISARD practitioners often cite the 1-tuple case as a trivial one because, under these conditions, the NNT is equivalent to a mask matching operation. This is only true in WISARD (or indeed any LNN) where the sites are restricted to binary values. If, however, the sites can take on real values then the matching takes places in a weighted Hamming space and this affords us with rather more sophisticated matching metrics.

In our coding scheme for a 1-tuple system, the Hamming space is weighted with respect to the amount of information conveyed by any single tuple. If the pixel has a high DC component across the data set then it is not considered during the matching phase. Conversely, if the pixels conveys a large amount of information then the weight of that pixel during matching is biased accordingly. This probably accounts for the improved performance of the coding over the mask-matching systems.

6.9. Discussion.

Training a neural network is, typically, a rather hit and miss affair because the solution is often sensitive to both the learning rate and the order of presentation of data. In a conventional neural network system, the surest way of guaranteeing that the net will learn the optimal features is to set the learning rate very low. This incurs long training times which are not viable for a practical system. A higher learning rate will cause the neural network to converge faster but may cause it to converge to a sub-optimal solution. Selecting the best learning rate is no simple matter and usually requires an expert with recondite knowledge of the system dynamics. This is simply not practical for a system which is to be operated with only a modest level of expertise.

In an attempt to circumvent this difficult parameter selection phase, we have developed an entirely new and 'user-friendly' neural network learning law. It does not require a learning rate to be set, is insensitive to the order of presentation of the data set, and will always converge to a near-optimal solution.

We have shown that this learning law, though quite different in form, produces exactly the same transformation as the learning law developed by Sanger*[41]*, outlined in chapter 3 of this thesis. Sangers learning law makes several approximations which become increasingly invalid as the learning rate grows; this has been outlined by Oja*[54]*. However, no such approximation are made in the derivation of this learning law and, as a consequence, it does not suffer from this drawback. We show that the performance of these two nodes is remarkably similar for a database of machine printed fonts.

We showed how a variant of this learning law could be supported by both an n-tuple and a sum-of-weights type node.

As the number of nodes is increased in an n-tuple implementation, the expanded feature space afforded by larger tuple sizes can de-correlate the information conveyed by successive nodes. This was discussed in Section 5.9. This offers the prospect of improved performance over sum-of-weights type networks, though there are no apparent ground rules for matching the tuple size to the nature of the problem (number of nodes, type of data being coded etc.). In addition, n-tuple systems suffer the following draw-backs:

- The memory requirements of an n-tuple system are 2^n times that of a simple sum of weights type node.

- The n-tuple requires a binary input. Consequently grey-level images must be thresholded before being presented to the network. This operation typically destroys some of the information conveyed by an image. Note that binary data was used for the tests throughout this section and this shortcoming was not a problem.

Results presented in this section indicate that n-tuple nodes converge much faster than their sum-of weights counterparts.

Our scheme codes each image with respect to a set of features learnt by the neural network. These features characterise the distribution of images contained within the database. As images are added to the database then these features may change and the neural network may require re-training to optimise the performance of the retrieval mechanism. We show that if the images to be added to the database are similar to the class of images already contained within the database then there is no marked decrease in performance and that training is not particularly pressing.

Conversely, if the images are significantly different from those already contained within the database then the performance of the retrieval mechanism is degraded. In this case training will improve the performance quite considerably.

The optimum number of nodes depends upon the diversity of the images within the database. Increasing the number of nodes will generally improve the performance of the system, though the law of diminishing returns comes into play here; the improvement afforded by additional nodes decreases to the point where the addition of further nodes has a barely tangible effect. The standard deviation of the node output can be used to optimise the number of nodes within a system.

For the sum-of-weights type neural networks more nodes usually means longer training times. It is interesting to note that training times for the n-tuple node are remarkably insensitive to the number of nodes.

Training times are also dependent upon the number of images within the database; as the database becomes more extensive then training times become longer. Training times for n-tuple nodes are considerably less than sum-of-weights types nodes for relatively large databases. However, as the database becomes increasingly diverse the sum-of-weights type nodes begin to outperform n-tuple implementations.

As the database becomes larger and more diverse, the improvement afforded by additional training continues to diminish. In fact, for very diverse databases, the features learnt by the node stay remarkably constant and do not change appreciably. This is because, for a highly diverse set of images, the nodes begin to learn image primitives which characterise a very broad range of image types. Such primitives might include oriented line segments, for example. This has been noted by a number of researchers *[56][80][81][82]*. In fact, the second order statistics afforded by PCA, which take both spatial and ensemble correlations into account, become less and less significant as the distribution of the data set becomes more diverse. Eventually, the performance of PCA approaches that of a transform which considers only first order spatial statistics. These first order transforms, such as DCT or the Hadamard Transform, are considerably easier to calculate than PCA.

Many researchers have shown that the features learnt by PCA type nodes trained on a diverse set of images resemble the so-called receptive-fields found to exist within mammalian processing systems . This seems to give some biological plausibility to our own approach. However, the fact that such features remain constant for highly diverse databases (the mammalian receptive fields are similarly fixed) raises an interesting issue. Once we know what these features are, do we really require the learning mechanism provided by the neural network?

Our work indicates that for a completely unconstrained database, a learning mechanism is not strictly required once we know the form of the so-called receptive fields (whether it be the contents of the RAM in the case of an n-tuple system or the weight vector in a sum-of-weights type node). In this case the image code could be found from the output of a node with fixed weights. This is rather gratifying as the training of neural networks is fraught with problems, not least being the time taken to train the system. This prospect opens up several very exciting avenues of research which are outlined in the next section.

Chapter 7

Discussion and Conclusions.

7.1. Defining the Problem

The broad objective of this work has been to achieve retrieval of images from large unconstrained databases using image content. The problem is typified by the need to locate a target image within a database where no numerical indexing terms are available. Here, retrieval is based upon important features within in an image and uses sample images or user sketches to specify a query. A typical query might be framed as "Find all images similar to this one", for example.

Existing methods to retrieve images from such databases rely on either a manual search of the entire data-base or an automated search of manually (or interactively) entered image descriptors. Both of these techniques require considerable human input; in the former this overhead is incurred every search session, and in the latter each time an image is added to the database. Such methods are extremely time consuming and this has had a profound effect on both the appeal and feasibility of Image Database Systems for typical end users - for most they are simply not a practical prospect. This type of application is by no means uncommon and potential users include: office and library applications, printing, publication and advertising, security, medicine, Geographic Information Systems, education and training, arts archiving, entertainment and broadcasting. These communities produce large amounts of visual information and have all expressed a need for flexible, convenient and robust retrieval mechanisms.

Image retrieval by content will only become a practical prospect when general purpose automatic feature extraction is possible. This relies heavily upon computer vision and pattern recognition to formulate efficient indexes. However, extracting indexing features has proved to be an extremely difficult process to automate using conventional image processing techniques. The aim of this project has been to show how neural networks can provide a practical, flexible and robust solution to this difficult problem.

The retrieval mechanism developed here is intended for large, non-specific databases. Retrieval from fairly small databases is probably best tackled by implementing efficient browsing mechanisms which allow the user to peruse sets of 'thumbnail images' (reduced versions of size which can be simultaneously displayed on a screen) efficiently rather than retrieval-by-content schemes.

Neither does our retrieval scheme attempt to address the problems associated with the more subjective type of user query. Such queries might be framed as "Retrieve all images of Hitler addressing crowds". A system to support this class of request is beyond the scope of this research and one must caution against over optimistic expectations for subjective image retrieval in the near future.

7.2. Existing and Previous Work

The general area of image retrieval is now receiving considerable attention world wide. The recent surge in the volume of research papers covering this type of application clearly attests to the growing interest in this embryonic and active area.

Image retrieval by content is being investigated using colour signatures and shape descriptors of discrete objects within a scene, for example *[11][12][13][20]*. The latter technique suffers from the need for interactive object detection using manual front ends. Colour signatures are not robust as images with identical signatures can have markedly different scene content. Contending methodologies include graph matching based on feature extraction of cloud images*[83]* and satellite images of the upper atmosphere*[20]*, for example. Neither of these technique attempt automatic feature extraction and both are specific to very narrow classes of problems.

The main comments on the current position regarding Image Retrieval *[2][4][78][84]* are:

- There is a significant and on-going requirement for image retrieval and, whilst existing systems are targeted at various specific problem domains, there is a need for a more general purpose and robust approach.
- It is highly improbable that existing strategies (i.e. colour signatures and shape descriptors) will scale to more general applications.

- Neural Networks, as demonstrated in this report and by other researchers *[44]*, offer the prospect of a more robust and flexible image coding strategy.

7.3. Basic Thrust of the Neural Network Coding Scheme.

Neural networks consist of large arrays of comparatively simple processing elements, or 'neurons', which interact in parallel to learn a mapping from a problem to a solution in a non-deterministic manner. The thrust of this project was to develop Neural Network Transforms (NNT's) for image retrieval. The neural network represents each image with respect to a number of feature archetypes learnt during a training phase. These archetypes characterise the distribution of images within the database and exist as a points in multi-dimensional feature space. The index represents the length of the vector between the image and the archetype.

The array of neural networks is trained on the entire database until convergence. Each image is then presented to the networks and the response of each node taken as an element of the index vector. The Neural Network Transform, from image to index, preserves the relative relationships between the images in the original domain so that the index can be used as the basis of the similarity metric. High dimensional images can be compared through their low dimensional codes. Images whose codes have the shortest Euclidean distance between them may be deemed to be most similar in the original pattern space and this forms the basis of the retrieval mechanism.

This research area is clearly being approached from a 'bottom-up' strategy. The initial successes will be with very specific databases and will then move towards the more general area of image retrieval. Our own approach to date has been centred on global, 'full-scene' features for the image indices for the retrieval and matching mechanism. We believe that the techniques developed to date could be adopted for formulating indices from more local features in due course.

Local features provide more comprehensive and robust scene descriptors and have the potential to support a degree of invariance to translation, scale and occlusion, for example. However, practical methods to embody these comparatively complex scene descriptors within a suitable data structure to support efficient matching have yet to surface. Global descriptors are not afflicted with this drawback and much of their efficacy may be attributed to their underlying simplicity.

7.4. What Kind of Neural Transformation Do We Want ?

Our matching metric requires that images close together in the pattern space yield correspondingly similar codes in the code space. This criteria is optimally satisfied by a single layer linear NNT.

However, the neural network community have, by and large, tended to regard such architectures with low esteem because of their comparatively limited functionality. This scant regard for the capabilities of the humble linear neuron has arisen because most neural network applications have tended to be centred around multi-layer classifier systems trained through unsupervised learning, as in the ubiquitous backpropogation training algorithm for multi-layer perceptrons, for example. In this context it is true that the linear transformation is somewhat lacking in that it will not cluster data into notionally distinct classes and the fact that multi-layer linear topologies do not provide any computational advantage. However, unlike a non-linear NNT, its linear counterpart does have a distance preserving property which is more important for unsupervised learning systems. Theoretically, a linear NNT will capture most information within a data set *[56]* (by contrast, a non-linear NNT, will destroy some information). Whilst the work carried out in this project corroborates this, it does indicate that a small amount of non-linearity during training can speed up the learning process considerably and still preserve the relative distances between the images in the original domain.

7.5. Developing a Suitable Learning Law for the Coding Scheme.

We have shown how a single layer linear neural network and accompanying learning law can be used as the basis for our coding scheme. The learning law investigated initially was developed by Sanger*[41]*, Oja*[54]* and Linsker*[56]* and is directly equivalent to Principal Component Analysis (PCA). PCA is a statistical analysis technique which isolates salient characteristics within the data set and enables it to be represented in a lower dimensional bound. The classical realisation of PCA is very computationally expensive and not suited to high dimensional data sets but can be implemented very efficiently on neural network architectures. In this context the neural network is acting as an information filter. However, the learning laws, developed by Sanger *et al* are only an approximation of PCA.

The accuracy of the approximation is determined by the rate at which learning is undertaken within the system. If the learning rate is small, the approximation is accurate, the weights will converge to the Principal Components of the data set but will take a long time to get there. If the rate is high, convergence will be quicker but the approximation less accurate and the resultant transformation may not be optimal. The approximations inherent in the derivation of the learning law means that the learning rate can have a profound effect upon the performance of the code.

The size of the learning rate determines the extent of localisation of the learning process within the training ensemble. A small learning rate will 'spread' the context of the learning across the data set. We show that the learning rate should be made inversely proportional to the size of the data set.

We also show that the learning rate should be made inversely proportional to the size of the images within the database. The size of the images does not effect the performance of the code.

Training a neural network is, typically, a rather hit and miss affair because the solution is often sensitive to both the learning rate and the order of presentation of data. Selecting the best learning rate is no simple matter and usually requires an expert with recondite knowledge of the system dynamics. This is simply not practical for a system which is to be operated with only a modest level of expertise.

In an attempt to circumvent this difficult parameter selection phase, we have developed an entirely new and 'user-friendly' neural network learning law. It does not require a learning rate to be set, is insensitive to the order of presentation of the data set and will always converge to the Principal Components very rapidly.

We have shown that this learning law, though quite different in form, produces exactly the same transformation as the learning law developed by Sanger *et al.* However, no approximations are made in the derivation of this learning law and, as a consequence, it does not suffer from the drawbacks mentioned earlier. We show that the performance of the two learning paradigms is remarkably similar for a database of machine printed fonts.

Successive nodes learn features with decreasing significance. For a given data set there comes a point where the inclusion of additional nodes provides little or no additional information. The optimum number of nodes depends upon the diversity of the images within the database. Increasing the number of nodes will generally improve the performance of the system. The improvement afforded by additional nodes decreases to the point where the addition of further nodes has a barely tangible effect on the performance of the retrieval mechanism. For a database of moderately diverse image types (so that the distribution tends to that of a Gaussian) the standard deviation can be used to select the optimal number of nodes which occurs where successive nodes exhibit standard deviations of the same order of magnitude.

For the sum-of-weights type neural networks more nodes usually means longer training times. We have shown that, for a database of unconstrained image types, training times increase approximately linearly with the number of nodes. Training times are also dependent upon the number of images within the database; as the database becomes more extensive then training times become longer.

Summing up then, we can see that a simple single layer linear (or near linear) neural network can be used as the basis of our coding mechanism. The transform provided is equivalent to PCA but is less expensive both in terms of memory and computation time.

7.6. Logical Neural Networks for Coding Images.

The relative popularity of LNN architectures in the UK may be attributed to the early successes of the WISARD system. Indeed, the appeal of these structures is peculiarly parochial and LNN's are regarded as something of a 'dark horse' by researchers outside of the UK, who, by and large, tend to focus their research interest on sum-of-weights type nodes. WISARD is such an attractive prospect because it stands alone in its capability for rapid training and ease of implementation in hardware. Unlike sum-of-weights nodes it requires only one pass through the training set during training and can be readily scaled up to very large problems. WISARD can be trained on video images at video frame rates - no other current neural network architecture can match this performance.

For a single node system, a 1-tuple discriminator, which produces a linear NNT, will outperform larger tuple systems. This is because larger tuple sizes exhibit increasingly non-linear transfer characteristics and will not produce the conformal mapping required to preserve the relative distances between the images in the pattern space.

A Logical Neural Network will perform best if the tuples are mapped to completely un-correlated areas within the image retina. Normally, the tuples are mapped randomly onto the retina to achieve this. We present a novel mapping strategy which can improve upon random mapping. Here the tuples are mapped onto the retina such that the mean Euclidean distance between the tuple sites is maximised.

As the number of nodes is increased in an LNN, the expanded feature space afforded by larger tuple sizes can de-correlate the information conveyed by successive nodes. As the number of nodes grows the increased feature space afforded by large tuple sizes becomes more important than the conformal mapping produced by the smaller ones. This offers the prospect of improved performance over sum-of weights type networks, though there are no apparent ground rules for matching the tuple size to the nature of the problem (number of nodes, type of data being coded etc.).

WISARD is classically trained through unsupervised learning where one pattern class is assigned to each discriminator. Following this methodology, we initially experimented with training single patterns from within the database into individual discriminators by shifting and adding noise to the original. A key issue here was to find the most appropriate training pattern (and training strategy) which optimised the performance of the code.

The standard deviation of the node activity provides a rough metric with which to judge both the best training archetype and the extent of training required for any particular discriminator, though the result may be sub-optimal. The processing time for this scheme grows linearly with the number of nodes and as a square of the size of the database. However, because they can be easily be implemented in hardware LNN's can be trained extremely rapidly and this shortcoming is not as serious as it would be for other neural network models.

We introduce novel Self Organising learning strategy for LNN's which is functionally identical to the paradigm developed for sum-of-weights type nodes mentioned earlier. Training times for this learning law are remarkably insensitive to the number of nodes and are considerably less than that required by sum-of-weights types nodes for relatively large databases. As the database becomes increasingly diverse the sum-of-weights type nodes begin to outperform LNN implementations.

7.7. An Objective Appraisal of LNN's for our Coding Scheme.

The novel learning law (developed as an alternative to Sanger's learning paradigm for sum-of-weights type nodes) was initially intended for use in LNN's. However, since it became apparent that the law could be supported readily on sum-of-weights type nodes, we were forced to evaluate the efficacy of LNN's in the light of our own particular application. An objective appraisal of their applicability to our coding scheme might be helpful here.

The beauty of the WISARD system lies with its simplicity. When more sophisticated training algorithms are implemented on LNN structures however, their advantages, most notably their ease of implementation in hardware and rapid training times, are gradually eroded. It is easy to loose sight of this fact and to stick doggedly to LNN's even though they may no longer be readily supported in hardware nor well suited to the problem at hand.

Our particular coding scheme requires a Self-Organising learning law. In such systems, the context of each pattern to the complete data set is particularly important. This can only really be implemented through multiple presentations of the data set. The 'one-shot' learning process employed in classical WISARD systems is rather crude since worthless features are given the same weight as important ones. This is not the case with more conventional neural network models where the relationship between any single training pattern and its context to the entire training set is taken into account over successive training presentations. In our particular application the data is not volatile since we can gain access to every pattern relatively easily during training. A more refined methodology, using multiple presentations of the data set in an 'off-line' training phase, would prove more effective than simple 'one-shot' learning here.

The relative importance of any one feature (such as a pattern of bits occurring within a single tuple) must be represented fairly explicitly for Self Organised learning. This requires that each site take on multiple values and is a characteristic of all Self Organising LNN's (in a WISARD system the site values are usually binary). Since the memory requirements of an n-tuple system are 2^n times that of a simple sum of weights type node, LNN topologies for Self Organising systems, require comparatively large amounts of memory.

The learning capacity in Self Organising systems must be regarded as a fixed resource - the very act of learning something means that another, less important, item of data must be relinquished. As a consequence, some scanning of the feature space is required in order to normalise the learning resource in LNN's. Since the feature space grows exponentially with tuple size, this scanning operation can incur considerable overheads during training. Fortunately, this normalisation phase is implicit in the novel Self Organising LNN learning law developed here and no scanning of the feature space is required. This is the only Self Organising learning law for LNN's which does not suffer this drawback.

One of the great advantages of n-tuple systems is that higher tuple sizes produce increasingly non-linear transforms which can help to cluster the data in classifier systems. However, for small numbers of nodes, our coding scheme works best where the NNT is linear (i.e. a 1-tuple) and the non-linearities produced by larger tuple systems are not particularly advantageous. Larger tuple sizes can help when a large number of nodes are used but matching the tuple size to the nature of the problem domain can only be performed through 'trial and error'. There is no apparent concrete relationship between the tuple size, the standard deviation of the node output and the performance of the retrieval mechanism.

Though much work has been undertaken on applying LNN's to grey-level data, these architectures really come to the fore with binary data. Unfortunately, thresholding grey level images will, typically, destroy large amounts of information within the image. This is undesirable. Sum-of-weights type nodes can accept grey level data and this is a distinct advantage for our particular application.

Sanger shows that a multi-node training scheme can be implemented by forcing successive nodes to learn orthogonal features. Orthogonal learning is induced by removing fractions of the features learnt by earlier nodes from the input to successive ones. Such an orthogonalising strategy cannot be easily implemented on LNN's because the input must be binary. Thus we are forced to adopt the more crude 'winner-take-all' type learning scheme which can impair the performance of the code.

WISARD is usually used as a classifier and trained through supervised learning. However, our own requirement necessitates the use of unsupervised learning and comparatively limited research has been undertaken in this area for LNN's. It is not apparent that the very attributes which render WISARD such an attractive prospect for are at all applicable for our coding system (a dimensionality reducer trained through unsupervised learning). In conclusion, this author feels that sum-of-weights type nodes are more suited to our coding scheme than LNN's.

7.8. When do we Need to Retrain the System?

The NNT optimises the performance of the code with respect to the changing contents of the database. As new images are added to the database there will come a time when the nodes need to be re-trained, although the degradation in performance is gradual and, where the database contains a broad range of image types, almost imperceptible. The need for re-training is dependent upon the changing distribution of images in the pattern space.

Our coding scheme works best when there is a theme which runs through the database, such as a database of facial images, for example. The more specific the theme, the greater the 'computational advantage' of our NNT.

We show that if the images to be added are similar to the class of images already contained within the database then there is no marked decrease in performance and that training is not particularly pressing. Conversely, if the images are significantly different from those already contained within the database then the performance of the retrieval mechanism is degraded. In this case training will improve the performance quite considerably.

The Neural Network Transform attempts to extract second order statistics from the data set. These statistics take both spatial and ensemble characteristics into account in an attempt to represent the images in a lower, and so more manageable domain.

As the database becomes larger and more diverse, the improvement afforded by additional training continues to diminish. In fact, for very diverse databases, the features learnt by the node stay remarkably constant and do not change appreciably. This is because, for a highly diverse set of images, the nodes begin to learn spatial image primitives which characterise a very broad range of image types. Such primitives might include oriented line segments, for example. In fact, the second order statistics afforded by PCA, which take both spatial and ensemble correlations into account, become less and less significant as the distribution of the data set becomes more diverse. Eventually, the performance of PCA approaches that of a transform which takes into account only first order spatial statistics. These first order transforms, such as the Discrete Cosine Transform, are considerably easier to calculate than PCA.

We have used global features for our current coding scheme where each neuron covers the entire image. However, a neuron can be trained on much smaller localised patches within each image. When trained with a broad range of image types such neurons begin to attune to local feature primitives which are almost identical to the 'retinal fields' found to exist within mammalian visual systems. These fields, located within the primary visual cortex, are used to encode images for subsequent processing centres 'higher up' in the cortex. This degree of biological plausibility, to a considerable extent, validates the approach that has been adopted here and provides evidence of a long-term solution to the problem since retrieval by content is clearly a function of human information processing. Other researchers have proposed the use of such fields for image coding systems *[41][42][56][85]*. The fact that such features remain constant for highly diverse databases (the mammalian receptive fields are similarly fixed) raises an interesting issue. Once we know what these features are, do we really require the learning mechanism provided by the neural network?

Our work indicates that for a completely unconstrained database, a learning mechanism is not strictly required once we know the form of the so-called receptive fields (whether it be the contents of the RAM in the case of an LNN or the weight vector in a sum-of-weights type node). In this case the image code could be found from the output of a node with fixed weights. This is rather gratifying as the training of neural networks is fraught with problems, not least being the time taken to train the system. This prospect opens up several very exciting avenues worthy of further investigation.

7.9. A Coding Scheme Based on Fixed Local Transforms

For large, diverse databases the learning mechanism afforded by our (localised) neural network is no longer required. Indeed, it would be more efficient to implement the NNT on an appropriate off-the-shelf, hardware based image processor card. Similar transforms, often referred to as 'wavelets', have been noted by other researchers *[42][79][80][82][85]*. We propose the application of these local transforms as the basis of our coding scheme.

These first order local transforms have several attractive features:

- Simplicity of implementation.
- Guarantee that performance is close to optimal for a large database.
- Local features permit scale invariance through a hierarchical processing structure and are better at isolating important component objects within a scene that global ones. Processing 'up' through higher layers enables extended features to be isolated. Thus, we can extract features at several resolutions which provides much scope for compact scene descriptors for the indices.
- Spatial filtering operations which convolve the image with a simple mask operator are very common in image processing applications. As a consequence, hardware capable of performing these mask operations extremely rapidly is available at a modest cost. Since our receptive field can be regarded as just such a mask then there is potential for a very rapid coding algorithms here.

We propose the following strategy:

i. Train a neural network on a very diverse set of (grey level) images to isolate a fairly extensive set of local receptive fields. Several banks of neural networks are trained on different resolutions of the images to isolate receptive fields for a number of processing hierarchies.

ii. The features learnt by the network are transferred to a mask which can be programmed into a proprietary hardware-based convolution processor. This processor will reside as a plug-in board within a PC, for example.

iii. Convolve each image with these masks, starting with the lowest resolution (i.e. the smallest). Record the output from each mask in each position of the image. Those masks with the largest output capture most information regarding localised image artefacts and may be deemed to be the most appropriate descriptor for that particular image entity.

iv. This convolution takes place at several scales, progressing up through the processing hierarchies to 'look' for larger local features. As soon as the output of the node for any region within the image is less than the weighted-mean of the mask output at the layer below then the movement up through the processing hierarchies stops.

v. The highest output (scaled with size) is recorded as a significant feature within a scene. An object or image entity may be defined in terms of the most significant receptive fields and the processing plane in which the processing 'stopped'. This plane number permits a degree of scale invariance.

We believe that a thorough investigation of correlation between the masks outputs will be necessary. This degree of so-called lateral interaction between the local receptive fields would permit more economic descriptions of the scene.

The hierarchical nature of the processing suggests that this type of system could be supported by an array of parallel processors, such as transputers, for example. A feasibility study needs to be undertaken here to ascertain the most appropriate 'off-the-shelf' processor card for our proposed PC based demonstrator.

7.10. Epilogue.

Neural networks are frequently regarded by the uninitiated as a kind of computational panacea. It is all too easy to get the impression that one merely has to 'throw' data at an off-the-shelf architecture, leave the neural network to 'get on with it' and come back a short while later to find the problem solved. In reality though, this is simply not the case and a neural network practitioner must typically ponder long and hard over a host of frequently baffling network parameters before even beginning to tackle the problem at hand. Such parameters might include network topology, nodal transformation function, learning law and its associated parameters such as the learning rate etc. This is not an easy task and the selection process has, in past, often been closer to alchemy than to science.

Neural networks do, however, offer a computational strategy for solving mathematically intractable problems where no analytical solution is likely to be forthcoming in the foreseeable future. Thus, they are an enabling paradigm for many problems such as the image retrieval task outlined here.

Neural networks have definitely fired the imagination of potential users but it is fair to say that they have yet to live up to their promise. Public confidence in neural network technology will match public curiosity when the current limitations of these devices are addressed in order to provide robust, efficient and, above all, practical solutions to real world problems. This will only come about with further research and ambitious, demonstrator-based projects.

Bibliography

1. Rickman. R. *'British Library Research and Development/Brunel University Report on the SPIE/IS&T Conference on Storage and Retrieval from image and video databases. San Jose, USA.Feb 1993'.* Feb. *1993.*

2. Jain R, Report on the National Science Foundation *'Workshop on Visual Information Management Systems'*, Redwood City, California, U.S.A. Feb. 24th to 25th, *1992.*

3. IEEE Computer Special Issue on *'Image Database Systems'* Eds. Dec. *1989.*

4. Chang. S. K, *'Image Information Systems: Where do We Go From Here ?'.* IEEE Trans. on Knowledge and Data Engineering. Vol. 4, No. 5, *Oct 1992.*

5. Salton. G, and McGill. M. J, *'Introduction to Modern Information Retrieval'.* McGraw-Hill. *1984.*

6. Tamura. H, and Yokota. N, *'Image Database Systems: A Survey'.* Pattern Recognition. Vol.17 No.1, pp.29 - 43, *1984.*

7. Chang. S. K, *'Pictorial Information Systems Design'* McGraw-Hill. *1989.*

8. Dyson. M. C. *'How do you describe a symbol ? The problems involved in retrieving symbols from a database'.* Information Services and Use. Vol. 12. pp. 65 - 76. *1992.*

9. Chang. S. K, Yan. C.W, Dimitroff. D.C. and Arndt. T,*'An Intelligent Image Database System'.* IEEE Trans on Software Enginerring. Vol SE14 No. 5 pp. 681 - 688. May *1988.*

10. Niblack. W, Barber. R, Equitz. W, Glasman. E, Petkovic. D, Yanker. P, Faloutsos. C, *'The QBIC Project: Querying Images by content using colour, texture and shape'* IBM Internal Research Report. IBM Research Division, Almaden Research Centre, San Jose, CA. USA. Feb. *1993.*

11. Niblack. W, Barber. R, Equitz. W, Glasman. E, Petkovic. D, Yanker. P, Faloutsos. C, *'The QBIC Project: Querying Images by content using colour, texture and shape'.* Proc. Int. Conf. on Storage and Retrieval for Image and Video Data Bases. San Jose, CA, USA. *SPIE/IS&T* Feb. *1993.* (In Print).

12. Arman. F, Hsu. A. Chiu. M. Y., *'Feature Management for Large Databases'.* Proc. Int. Conf. on Storage and Retrieval for Image and Video Data Bases. San Jose, CA, USA. *SPIE/IS&T* Feb. *1993.* (In Print).

13. Swain M, *' Interactive Indexing to Image Databases'.* Proc. Int. Conf. on Storage and Retrieval for Image and Video Data Bases. San Jose, CA, USA. *SPIE/IS&T* Feb. *1993.* (In Print).

14. Rickman. R and Stonham T. J. *'Similarity Retrieval form Image Databases - Neural Networks can Deliver'.* Proc. Int. Conf. on Storage and Retrieval for Image and Video Data Bases. San Jose, CA, USA. *SPIE/IS&T* Feb. *1993.* (In Print).

15. Grosky. W. I. and Jiang. *Z 'A hierarchical approach to feature indexing'.* Proc. Int. Conf. on Storage and Retrieval for Image and Video Data Bases. San Jose, CA, USA. *SPIE/IS&T* Feb. *1992.*

16. Gevers. T and Smeulders. A. W M, *'Σnigma: An Image Retrieval System'* Int. Conf. on Pattern Recognition (ICPR) Vol. 2 pp697 - 700. The Hague, The Netherlands. IAPR/IEEE. 1992.

17. Grosky. W. I. and Mehrortra. *'Index-Based Object Recognition in Pictorial Data Management'.* Computer Vision, Graphics and Image Processing. Vol. 52 No.3 pp.416-436. *1990.*

18. Mehrotra. R and Grosky. W. I. *'Shape Matching Utilizing Indexed Hypotheses Generation and Testing'.* IEEE Journal of Robotics and Automation. Vol. 5. No.1 pp. 70 - 77. *1989.*

19. Daneels D, Van Campenhout D, Niblack W, Equitz W and Barber R, *'Interactive Outlining: An Improved Approach'.* Proc. Int. Conf. on Storage and Retrieval for Image and Video Data Bases. San Jose, CA, USA. *SPIE/IS&T* Feb. *1993.* (In Print).

20. Samadini R, Han C, *'Computer Assisted Extraction of Boundaries from Images'.* Proc. Int. Conf. on Storage and Retrieval for Image and Video Data Bases. San Jose, CA, USA. *SPIE/IS&T* Feb. *1993.* (In Print).

21. Tamura. H, Mori. S. and Yamawaki. T,*'Texture features corresponding to visual perception'.* IEEE Trans. on Systems, Man and Cybernetics Vol 8 No. 6. pp. 460 - 4773. *1978.*

22. Wakimoto. K, Shima. M., Tanaka. S and Maeda. A. *'Content-Based Retrieval Applied to Drawing Image Databases'*. Proc. Int. Conf. on Storage and Retrieval for Image and Video Data Bases. San Jose, CA, USA. *SPIE/IS&T* Feb. *1993*. (In Print).

23. Mumford. D. *'Mathematical Theories of Shape: Do they model perception ?'*. Geometric Methods in Computer Vision. Vol. 1570. pp. 2 - 10. SPIE. *1991*.

24. Mumford. D. *'The problem with robust shape descriptions.'* Proc. First Int. Conf. on Computer Vision. pp. 602 - 606, London. U.K. IEEE. *1987*.

25. Kato. T, Kurita. T , Otsu. N. and Hirata. K, *'A sketch retrieval method for a full colour image database'*. Proc. IAPR International Conference on Pattern Recognition (ICPR). pp. 530 - 533. The Hague, The Netherlands. IAPR. Sept. *1992*.

26. Kato. T,*' Database architecture for content-based image retrieval'*. Proc. Int. Conf. on Storage and Retrieval for Image and Video Data Bases. San Jose, CA, USA. *SPIE/IS&T* Feb. *1992*.

27. Rickman R. M. and Stonham. T. J. *'Coding facial images for database retrieval using a Self Organising neural network.'* IEE Colloquium on Machine Storage and Recognition of Faces. London. Digest 1992/017. Jan. *1992*.

28. Rickman R. M. and Stonham. T. J. *'Image retrieval from large databases using a neural network coding scheme.'* in Jones. K. P.(Ed.) Proc. conf. Informatics 11: The Structuring of information. pp. 147 - 159. York. March *1991*.

29. Beckman. N, Kriegel. H. P, Schneider. R and Seeger. B. *'The R*-Tree: an efficient and robust access method for points and rectangles'. Proc.* ACM SIGMOD, pp. 322 - 331, May *1990*.

30. Gutman. A *'R-Trees: a dynamic structure for spatial searching'. Proc.* ACM SIGMOD, pp.47 - 57. June *1984*.

31. Jagadish. H. V. *'Spatial Search with polyhedra'*. Proc. Sixth IEEE Int. Conf. on Data Engineering. Feb. *1990*.

32. Samet. H. *'The design and Analysis of Spatial Data Structures'*. Addison-Wesley, *1989*.

33. Nievergelt. J, Hinterberger. H. and Sevick. K. C, *'The grid-file: an adaptable symetric multikey file structure'*. ACM TODS, Vol. 9 No. 1 pp.38 - 71. March *1984*.

34. Hunter. G. M and Steiglitz. *'Operations on images using quad-trees'.* IEEE Trans. on PAMI. PAMI Vol. 1. No.2. pp.145 - 153. April. *1979.*

35. Maren. A. J, Harston. C. T. and Pap. R. M, (Eds.)*'Handbook of neural computing applications.'*San Diego, Academic Press. *1990.*

36. Chock. M, Cardenas. A. F and Klinger. A. *'Database structure and manipulation capabilities of a picture database management system (PICDMS).'*IEEE Trans. Pattern Analysis and Machine Intelligence Vol. PAMI-5, pp. 484 -492. *1984.*

37. Klinger. A and Pizano. A. *'Visual Structures and databases'.* in Visual Database Systems pp. 3 - 25. Amsterdam. North Holland. *1989.*

38. Rabitti. F and Stanchev. P,*'GRIM_DBMS: a Graphical IMage Database Management System.'* in Visual Database Systems. Amsterdam, The Netherlands. pp. 415 - 430. North-Holland. *1989.*

39. Fierens. F, Van Cleyneenbreugle J. , Suetens. P and Oosterlinck. A *'A software environment for image database research'.* Journal of Visual Language Computing. Vol. 3 pp 49 -68. *1992.*

40. Kofkis. P, Karmirantzos. A, Kavaklis. Y and Ourphasnoudakis. S.*'Image archiving by content: An object oriented approach'.* Proc. SPIE Medical Imaging IV: PACS System Design and Evaluation. Vol. 1234, *1990.*

41. Sanger. T. D. *'Optimal Unsupervised Learning in a Single Layer Feedforward Neural Network'.* Neural Networks. Vol. 2 pp. 459 - 473. *1989*

42. Koendric. J. J and Doorn van, A. J.*'Receptive field families'.* Biological Cybernetics Vol. 63. pp.291 - 298.*1990.*

43. Enser. P. G. B. *'Query Analysis in a Visual Information Retrieval Context'.* Research Report. Department of Library & Information Studies. University of Brighton. U.K. *1992.*

44. Dowe. J. *'Storage and Retrieval in Multi-Media Environments'.* Proc. Int. Conf. on Storage and Retrieval for Image and Video Data Bases. San Jose, CA, USA. *SPIE/IS&T* Feb. *1993.*

45. McCulloch, W. S. and Pitts *'A Logical Calculus of Ideas Immanent in Nervous Activity'*. Bulletin of Mathematical Biophysics Vol. 5, pp.115-133.*1943*. Re-printed in Anderson and Rosenfeld *[86]*.

46. Rosenblatt, F. *'Principles of Neurodynamics'*. New York: Spartan. *1962*.

47. Minsky, M. L. and Papert, S.A. *' Perceptrons'*. Cambridge: MIT Press. *1967*. Partially re-printed in Anderson and Rosenfeld *[86]*.

48. Rumelhart, D. E. and McClelland, J. L. and the PDP Research Group. *'Explorations in Parallel Distributed Processing'*. Vol.1. Cambridge: MIT Press. *1986*.

49. Cottrell, G. W., Munro, P. and Zipser, D. *'Learning Internal Representations from Gray-Scale Images: An example of Extensional Programming'*. Ninth Annual Conference of the Cognitive Science Soc. Seattle. pp. 462-473. Hillsdale: Erlbaum. *1987*.

50. Baldi, P. and Hornick, K. *'Neural networks and Principal Component Analysis: Learning from Examples without Local Minima'*. Neural Networks Vol. 2. pp.53-58. *1989*.

51. Kirby, M. and Sirovitch, L. *'Applications of the Karhunen-Loeve Procedure for the characterisation of Human Faces'*. IEEE Trans. on Pattern Recognition and Machine Intelligence. Vol. 12 No.1 pp 72 - 73. *Jan 1990*.

52. Turk, M. and Pentland, A.*'Face Processing: Models for Recognition'*. in Proc. 8th SPIE on Intelligent Robots and Computer Vision.*1989*.

53. Oja, E. *'Subspace Methods for Pattern Recognition'*. Letchworth: Research Studies. *1983*.

54. Oja, E. *'A Simplified Neuron as a Principal Component Analyser'*. Journal of Mathematical Biology Vol.15. pp.267-273. *1982*.

55. Oja, E. *'Neural Networks, Principal Components and Subspaces'*. International Journal of Neural Systems. Vol. 1 pp.61-68. *1989*.

56. Linsker, R. *'Self-Organisation in a Perceptual Network'*. Computer. March *1988*, pp. 105-117.

57. Hebb. D. O. *'The Organisation of Behavior'*. New-York:Wiley. (*1949*). Re-printed in *[86]*.

58. Bourland, H. and Kamp, Y. *'Auto-Association by Multi-Layer Perceptrons and Singular Valued Decomposition'.* Biological Cybernetics Vol. 59. pp 291-294. *1988.*

59. Clark, R. J. *'Transform Coding of Images'* Academic Press. *1985.*

60. LeCun. Y, Boser. B., Denker. J. S. *et al 'Backpropagation Applied to Hand-written Zip Code Recognition'.* Neural Computation: Vol 1. pp. 541 - 551. *1991*

61. Bledsoe. W., Browning., *'Pattern Recognition and Reading by Machine'.* Proc. Eastern Joint Conference. pp. 225 - 232. *1959.*

62. Aleksander. I., Stonham. T. J. *'A Guide to Pattern Recognition using Random Access Memories'.* Computer and Digital Techniques. Vol. 2 pp 29 - 40. *1979.*

63. Gurney. K. N. *'Training Nets of Hardware Realisable Sigma-Pi Units'.* Neural Networks. Vol. 5. pp. 289 - 303. *1992.*

64. Aleksander. I. *'Weightless Neural Tools : Towards Cognitive Macro structures'.* CAIP Neural Networks Workshop. Rutgers University, New Jersey, USA . Oct. *1989.*

65. Ullman. J. R. *'Experiments with the N-Tuple Method of Pattern Recognition'.* IEEE Trans. on Computers. Dec. *1969.*

66. Kanerva. P. *'Sparse Distributed Memory'.* M.I.T. Press. *1988.*

67. Tattersall. G. P., Sixsmith. M. J. *'Speech Recognition using n-tuple Techniques'* . British Telecomms. Technical Journal. Vo.2 No. 8 pp. 50 - 60 . *1990.*

68. Watanabe. S. *' Knowing and Guessing - A Formal and Qualitative Study'.* j. Wiley and Sons.' *1969.*

69. Kerin. M. A., Stonham. T. J. *'A Self-Organising strategy for Digital Neural Networks'.* Journal of Intelligent Systems. Vol. 2 Nos. 1-4., pp. 291-311, *1992.*

70. Allinson. N. M., Brown. M. T. and Johnson. M. J. *' [0,1]^n Space Feature Maps - Extensions and Hardware Implementation.'* Proc. 1st IEE Conference on Artificial Neural Networks, London. pp. 261 - 264. *Oct. 1989.*

71. Tambouratzis. G., Stonham. T. J. *'Implementing Hard Self-Organising tasks using Logical Neural Networks'.* Proc. International Conference on Artificial Neural Networks (ICANN - 92), Brighton, U.K. Vol 1. pp.643-647. *Sept. 1992.*

72. Ntourntoufis. P. *'Self-Organising Properties of a discriminator based Neural Network'.* Proc. IJCNN, San-Diego, USA. Vol. 2. pp.319-324. *June 1990.*

73. Kerin. M. A., Stonham. T. J. *'Self-Organisation and Autonomous Learning in Logical Neural Networks'.* Ph.D. Thesis. Dept. Electronic Engineering, Brunel University, Uxbridge. *June 1991.*

74. Carpenter. G. A. and Grossberg. S *'A Massively Parrallel Architecture for a Self-Organising Neural Pattern Recognition Machine'.* Computer vision, Graphics and Image Processing. Vol. 37. pp. 54-115. *1987.*

75. Kohonen. T. *'Self-Organisation and Associative Memory'.* Springer Verlag, Hiedelberg. *1984.*

76. Kohonen. T. *' Self-Organised formation of Topologically Correct Feature Maps'.* Biological Cybernetics. Vol. 43 pp.59-69. *11984*

77. Allinson. N. M., Johnson. M. J. and Moon. K. J.*' Digital Realisation of Self-Organising Maps.'* Neural Information Processing Systems Vol 1. pp. 728 - 738. Eds. Touretzky. D. S. , Morgan Kauffman Publishers, *1989.*

78. Rickman. R. and Stonham. T. J. *'A Self-Organising Logical Neural Network for Binary Data.'* Proc. International Joint Conference on Neural Networks (IJCNN), Beijing, China, Vol. 2 pp. 745-752. *Nov. 1992.*

79. Rickman. R. and Stonham. T. J. *'A Novel Learning Law for a Self-Organising Logical Neural Network'.* Wieghtless Neural Network Workshop. York. U.K. *April 1993.*

80. Daugman J. *'Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression'.* IEEE Trans. on Acoustics, Speech and Signal processing,. Vol. 36. No. 7. July *1988.*

81. Unser. M and Eden. M *'Multiresolution Feature Extraction and Selection for Texture Segmenation'.* IEEE Trans. on Pattern Recognition and Machine Intelligence. Vol.11 No. 7. pp. 717-728. Jul. *1989.*

82. Hancock. P. J. B, Baddeley. R. J and Smith. L. S *'The principal components of natural images'.* Network Computations in Neural Systems. Vol. 3 No. 1. pp.61 - 70. *1992.*

83. Kitamoto. A., Zhou. C., Takagi. M., *'Similarity Retrieval of NOAA Satellite Imagery by Graph Matching'*.Proc. Int. Conf. on Storage and Retrieval for Image and Video Data Bases. San Jose, CA, USA. *SPIE/IS&T* Feb. *1993. (In print).*

84. Petrie. J. H *'An overview of image processing and image management systems and their application'.* British Library Research Paper No. 40. Series 621.38' 0414. *1988.*

85. Watson. A. B. *'The Cortex Transform: Rapid Computation of Simulated Neural Images.'* Computer Vision, Graphics and Image Processing. Vol. 39. pp 311 -327. Academic Press Inc. *1987.*

86. Anderson, J. A. and Rosenfeld, E., eds. *'Neurocomputing: Foundations of Research'.* Cambridge: MIT Press. *1988.*

APPENDIX A

Database for the 'Intelligent' Video editor.

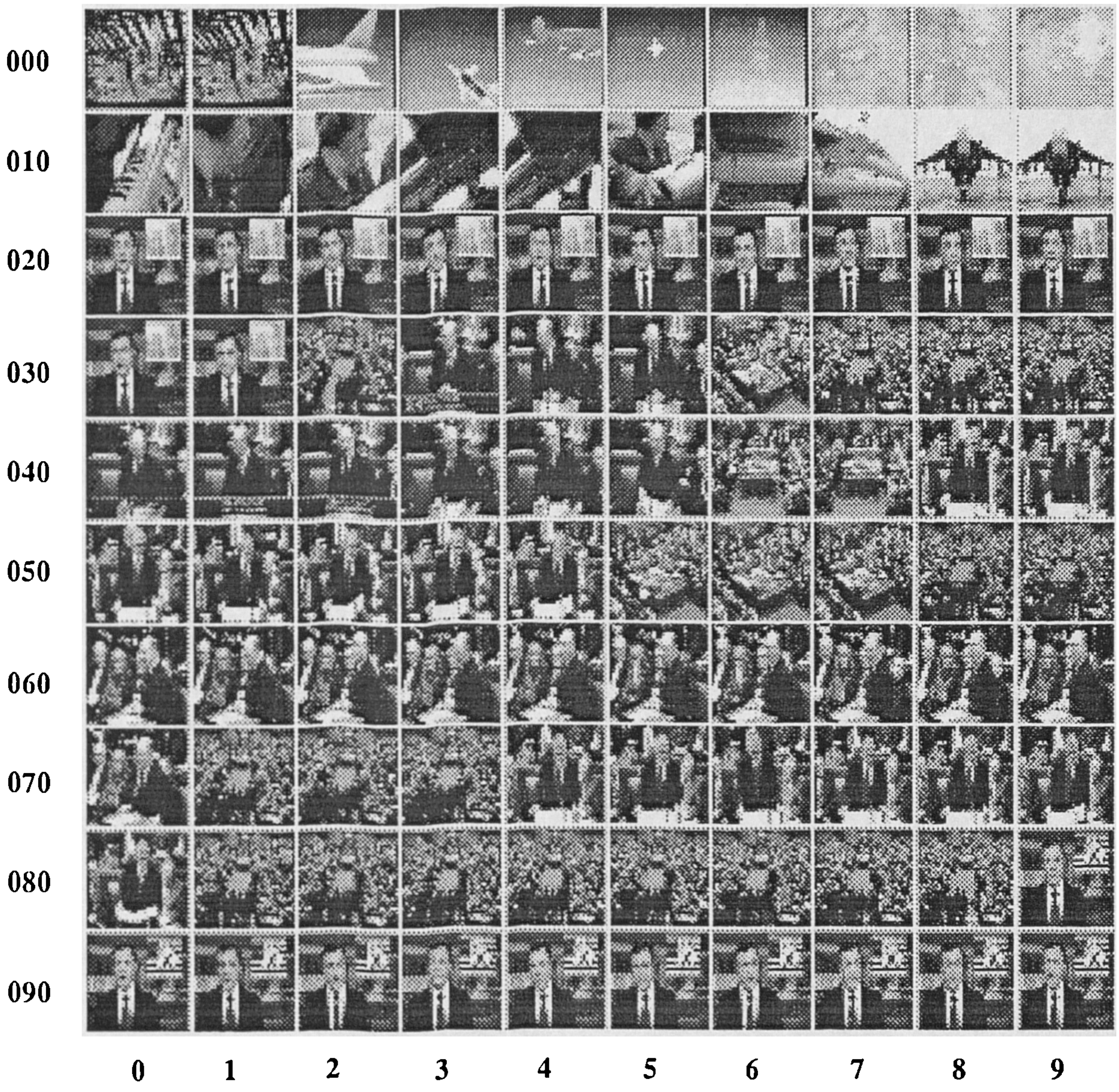


Fig. A.1 Images 0 - 99 in Video Database

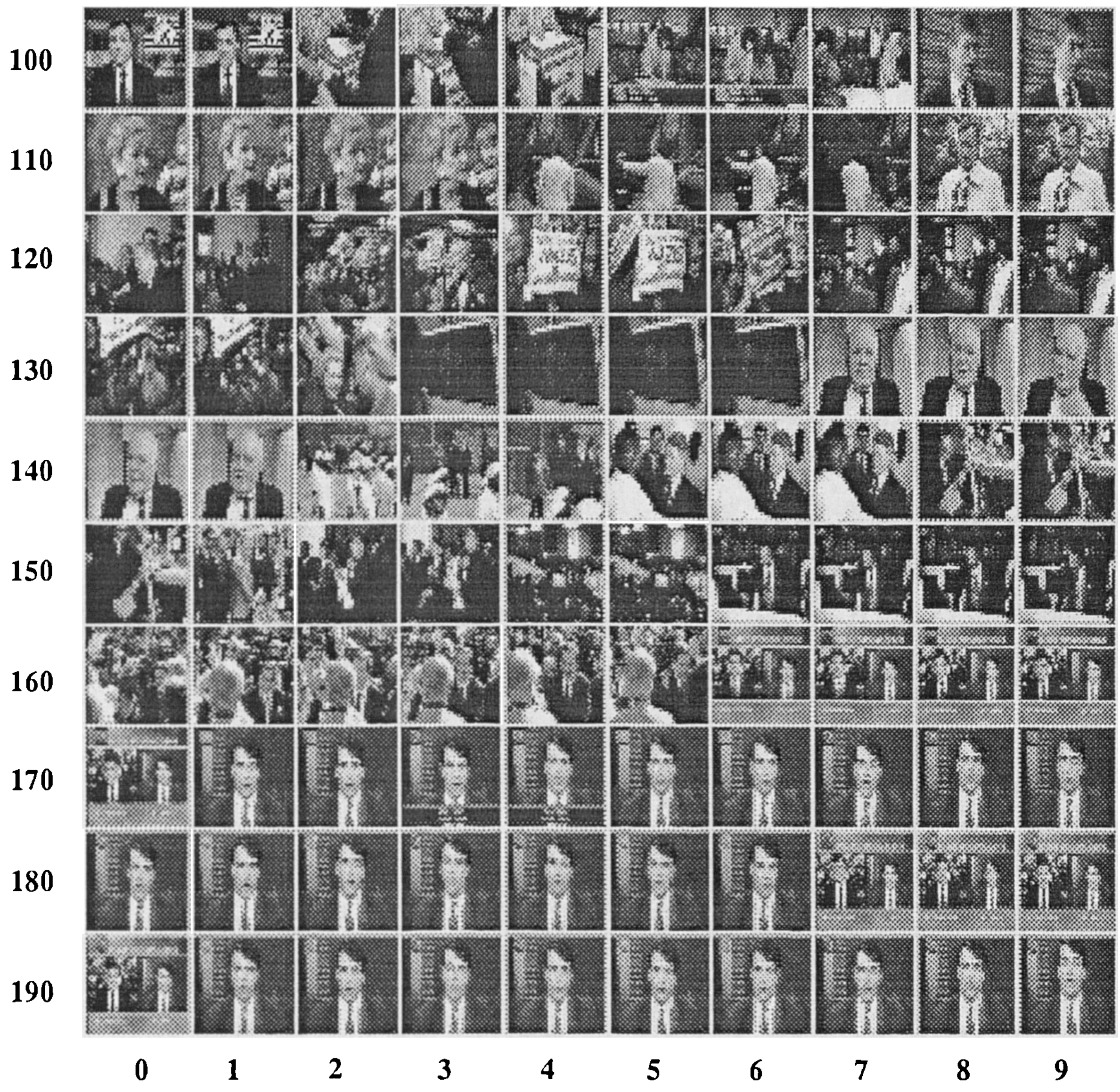


Fig. A.2 Images 100 - 199 in Video Database

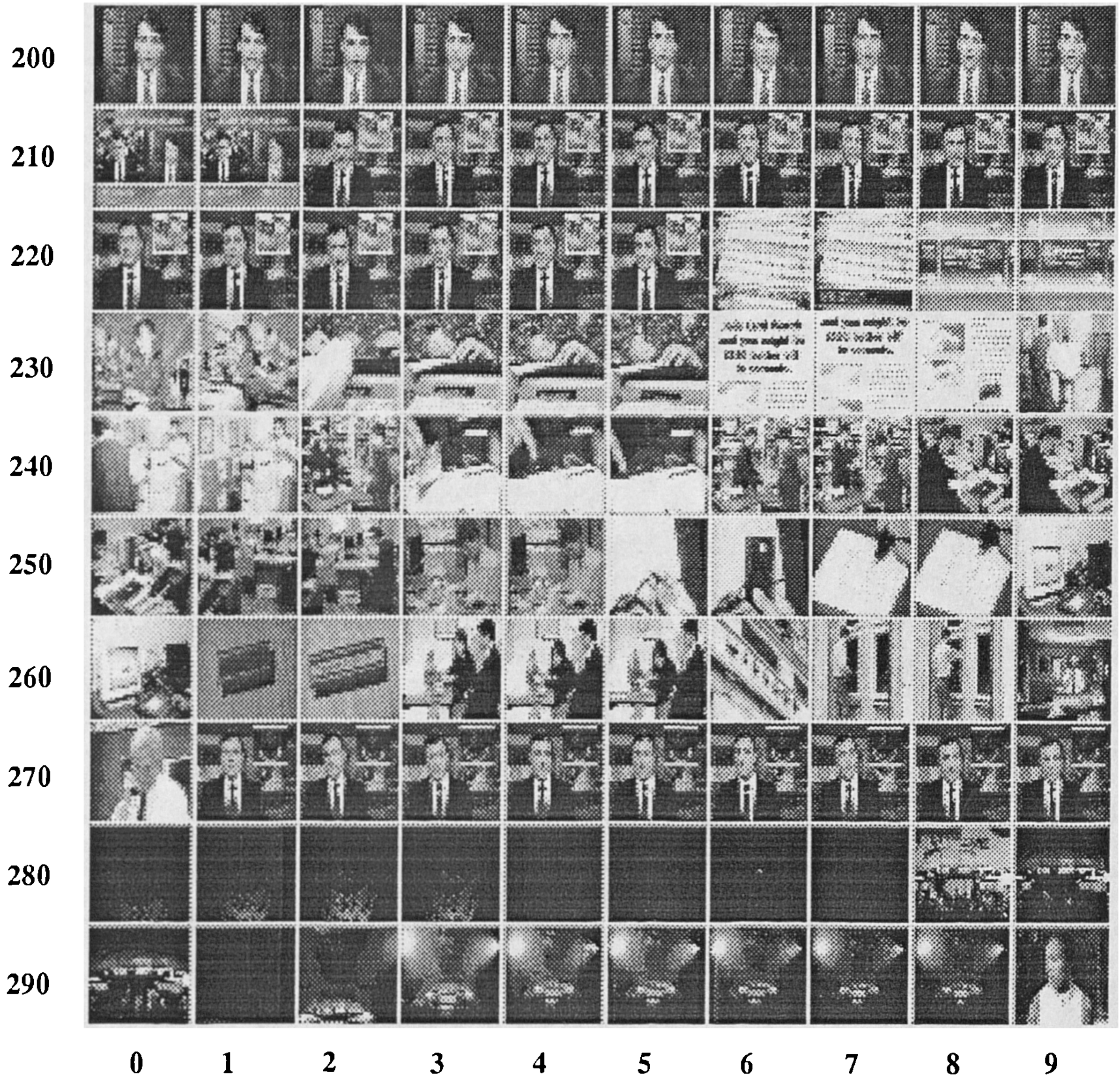


Fig. A.3 Images 200 - 299 in Video Database

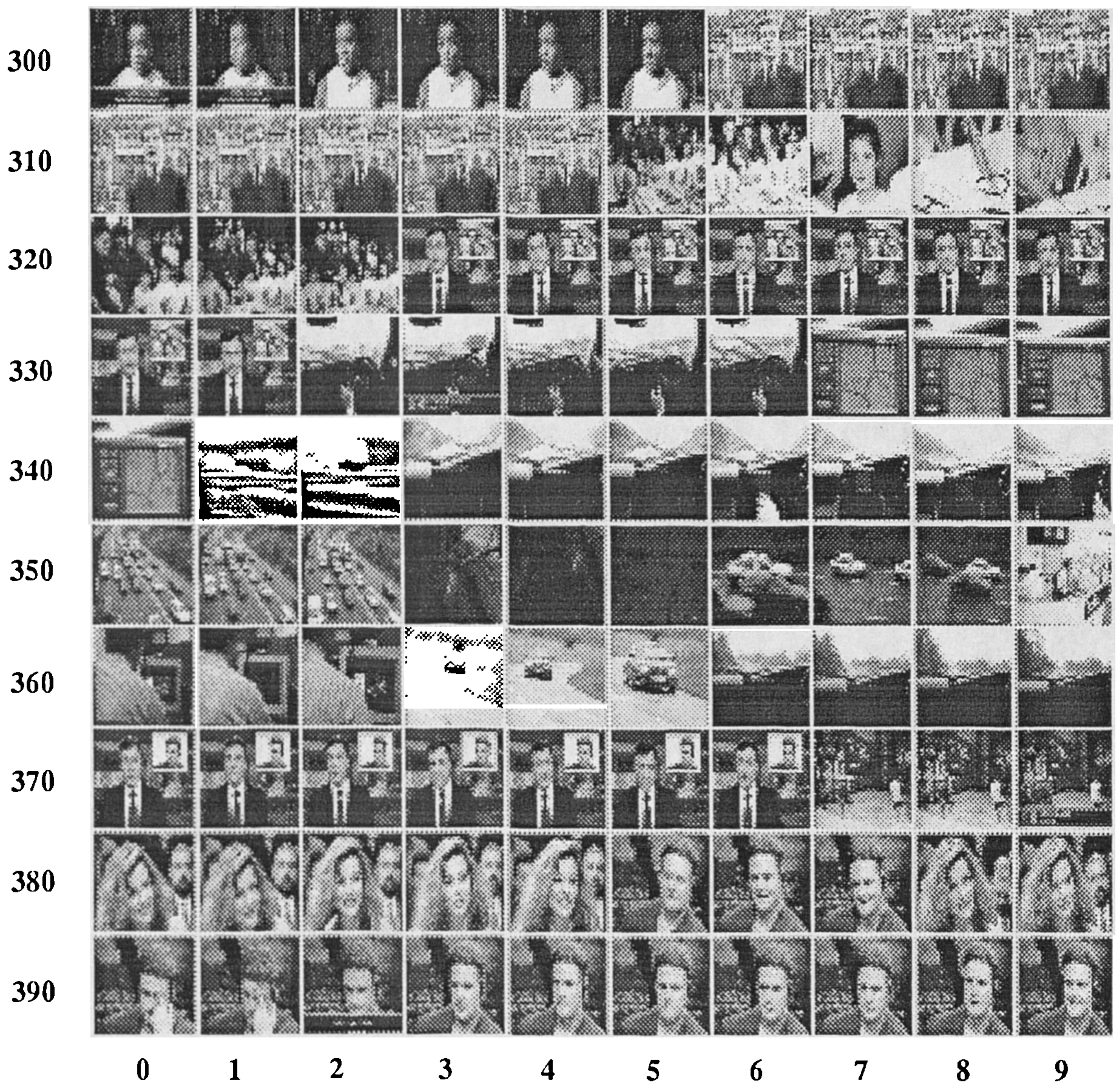


Fig. A.3 Images 300 - 399 in Video Database

APPENDIX B

A selection from the autors papers:

'A Novel Learning Strategy for Self-Organising Weightless Neural Networks'. Proc. Wieghtless Neural Network Workshop. York. U.K. *April 1993*......................................**167**

'Similarity Retrieval form Image Databases - Neural Networks can Deliver'. - Proc. Int. Conf. on Storage and Retrieval for Image and Video Data Bases. San Jose, CA, USA. *SPIE/IS&T* Feb. *1993*. (In Print)..**172**

'A Self-Organising Logical Neural Network for Binary Data.' - Proc. International Joint Conference on Neural Networks (IJCNN), Beijing, China, Vol. 2 pp. 745-752. *Nov. 1992*..**182**

'Coding facial images for database retrieval using a Self Organising neural network.' - Proc. IEE Colloquium on Machine Storage and Recognition of Faces. London. Digest 1992/017. Jan. *1992*..**189**

'Coding images for database retrieval using neural networks.' - Proc. 3rd NERVES Int. Workshop on Neural Networks. Grenoble. France. March. *1991*......................................**193**

A Novel Learning Strategy for Self-Organising Weightless Neural Networks

Rick Rickman and John Stonham
Neural Networks and Pattern Recognition Group
Department of Electrical and Electronic Engineering
Brunel University, Uxbridge, Middlesex UB8 3PH, UK.
e-mail: Richard.Rickman@brunel.ac.uk

Abstract

We present a novel Self Organising learning algorithm for weightless neural network architectures. The learning algorithm does not require a learning rate to be set and converges to the optimal solution very rapidly. In its simplest form, that is a 1-tuple implementation, the system is proved to converge to the Principal Components of the data set with probability 1. As the tuple size is increased the node performs progressively non-linear transformations of its input. We discuss the practical implications of such non-linearities and show how the learning law can be used as the basis of a coding mechanism for an image database retrieval system.

1 Introduction

Central to the thrust of any pattern recognition task is the notion of dimensionality reduction. The efficacy of a dimensionality reducing scheme rests with its ability to extract the salient features from an ensemble of data. The motivation behind the factors affecting the selection of such features has much in common with a classical statistical technique called Principal Component Analysis (PCA). Sanger[9],Oja[5][6] and Linsker[3] have shown that an approximation of PCA can be implemented very efficiently using a single-layer sum-of-weights type Self Organising linear neural network.

The update strategy for these architectures is based on an unsupervised Hebbian-type learning law which adapts the weights to correlate the input and output activity of the node. How can we implement such a learning law on a Weightless neural network architecture?

2 Hebbian Learning for Weightless Neural Networks

A simple 1-tuple system to support the Hebbian learning is shown in figure 1. It consists of an array of real valued memory elements which are addressed by the binary pattern on the input retina. Location 0 to each memory is addressed by a 'white' pixel and location 1 by a 'black'. The sites can take on positive or negative real values and are initially seeded by writing a '1' to those sites addressed by a pattern chosen arbitrarily from the training set.

The thrust behind Hebbian learning is to update the site commensurately with the strength of the node response. This can be achieved by adding the node response to the sites addressed by the training pattern. However, such a learning law will tend to accentuate the effect of those areas within the training data which have a large DC component and contain no discriminant information. This can be circumvented by forcing the mean of the node response to zero so that the total update for all of the sites addressed by a tuple is zero. Thus, a site which is addressed by every pattern will contain zero after one pass of the data set and will not contribute to the output of the node. This update strategy filters out unwanted DC components.

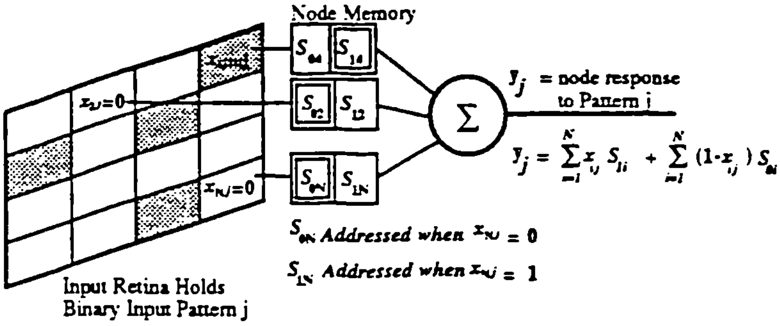

Node Memory
S_{0i}
S_{1i}
S_{02}
S_{12}
S_{0N}
S_{1N}
x_{ij}
$x_{2j}=0$
$x_{Nj}=0$
Σ
y_j = node response to Pattern j
$y_j = \sum_{i=1}^{N} x_{ij} S_{1i} + \sum_{i=1}^{N} (1-x_{ij}) S_{0i}$
S_{0N} Addressed when $x_{Nj} = 0$
S_{1N} Addressed when $x_{Nj} = 1$
Input Retina Holds Binary Input Pattern j

Figure 1. Basic architecture for the Self Organising Weightless neural network.

After one pass through the data set the new site values become:

$$\hat{S}_{1i} = \gamma \sum_{j=1}^{P} (y_j - \bar{y}) x_{ij} \qquad (1$$

$$\hat{S}_{0i} = \gamma \sum_{j=1}^{P} (y_j - \bar{y})(1 - x_{ij}) \qquad (2$$

Where

$\bar{y}$ is the mean response of the discriminator to the entire data set.

P is the number of patterns in the training set.

γ limits the dynamic range of the sites. Empirical tests have shown that setting γ as:

$$\gamma = \frac{1}{2} \sum_{j=1}^{P} \| y_j \| \qquad (3$$

will limit the dynamic range of the site values so that $-1 \leq (S_0, S_1) \leq 1$.

The complete training cycle may be summarised as:

i. Clear the discriminator by setting each memory location to 0.
ii. Seed the node by writing a 1 to sites addressed by an image chosen arbitrarily form the database.
iii. Present each pattern to the discriminator and record the response.
iv. Clear down the discriminator as in stage i.
v. Re-present each pattern to the node and add the normalised, mean-corrected response to the site addressed by that pattern.
vi. Repeat stages ii. to vi. until convergence.

This learning rule has been proven to converge to the eigenvectors of the auto correlation matrix of the data set (i.e. its Principal Components) with probability 1 [8].

This update rule has some rather interesting features:

- Unlike classical PCA this technique not require direct computation of the auto correlation matrix nor its eigenvectors which is impractical for high dimensional data sets.

- Unlike existing sum-of weights type neural network implementations of PCA. no approximations are made in the derivation of the leaning rule [4]. The learning rule needs no explicit learning rate and convergence is faster and more accurately.

- The learning rule does not implement the system updates until all of the data has been presented and will not converge to localised solutions dependent upon a particular ordering of the dataset.

- In many Self Organising Weightless neural network systems several sites within every n-tuple must be scanned on each training presentation to normalise the response of the node [1][2][4]. This can be time consuming. Normalisation in this learning law is implicit in the update mechanism and only one site need be accessed within each n-tuple on each training presentation.

After one pass through the data set $S_0+S_1= 0$ for a 1-tuple system. Thus. the memory requirements of this system could be halved simply by considering updates on the S_1 sites only. However. the notion that the white and black elements address distinct memory locations allows the learning law to be applied to systems with larger tuple sizes. Although the non-linearities induced by these higher order systems render the proof of convergence somewhat incomplete for tuple sizes greater than 1. the system dynamics are. ostensibly, still the same.

3 Does the code preserve the relationships between the images in the Pattern space?

Consider a training set of 16x24 bit binary characters. shown in figure 2 in which an 'A' font is gradually, and evenly, merged to a 'B' over 25 images. Image index '12' lies as half way between an 'A' and a 'B. A node trained to extract most information from the data will give a maximum equal and opposite response for image indexes 0 and 25 and zero response to index 12. Since the images are merged linearly the code produced by the node output should also change linearly from image index '0' through to '25.

The node was trained using the learning algorithm until convergence (deemed to occur when the respective node responses to each pattern differ by less than 0.1% of their former value). The results are shown below in figure 2.

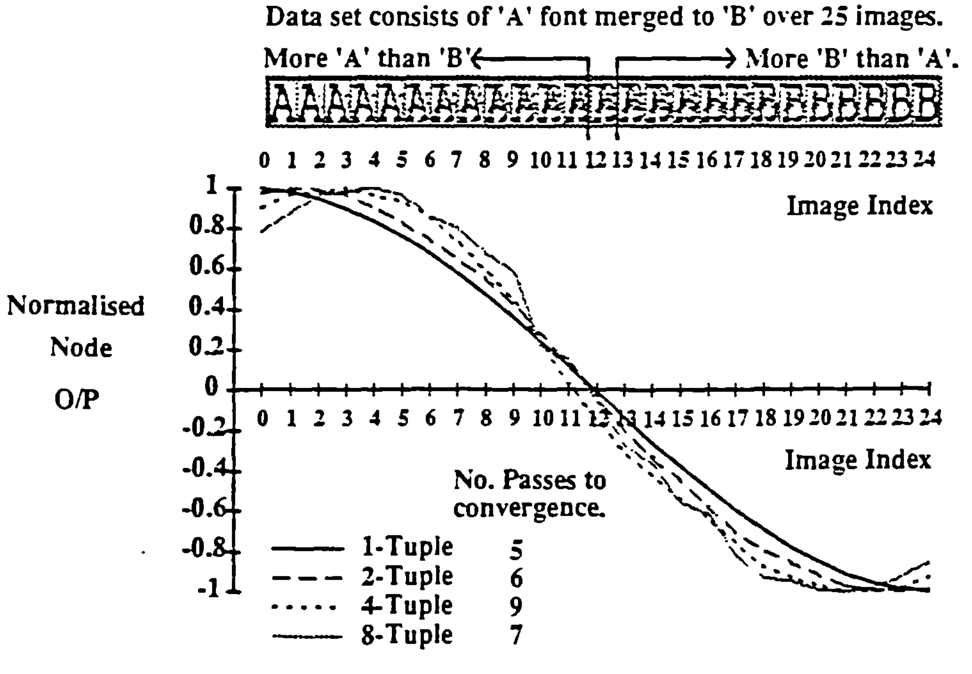


Figure 2. *Variation of node response with tuple size for a single node Weightless Neuron.*

A 1-tuple node behaves well as dimensionality reducer and its near linear response to successive images shows that it preserves the relative distances between the patterns in the original domain. As the tuple size increases, the node begins to behave more like a classifier.

4 Extending the learning law to multi-node systems.

Successive nodes in a multi-node Self Organising neural network should learn orthogonal features. In a 'winner-take-all' type learning scheme nodes compete for learning stimuli and orthogonality is achieved by virtue of the fact that a single node will learn a feature type specific to that node and no other. The automatic normalisation implicit in the learning law prevents the range of feature types learnt by that node from becoming excessively broad (where the same node responds maximally to each and every input image). Apart from this restriction, the learning law is identical to that presented above, bearing in mind that, initially, each node must be seeded differently to induce competitive learning between the nodes.

5 A Practical Implementation of the Learning Law.

The learning algorithm developed here can be used as the basis of a neural network coding scheme for an image database retrieval by content system [7]. Each image is coded by presenting it to an array of nodes which have been trained to extract the salient features which characterise the spread of images within the database. The response of each node forms an element of the code. The images are transformed into a very succinct code which facilitates rapid fuzzy matching of images within the data base.

The following test provides an objective measure of the performance of the retrieval by content mechanism. The data set, shown in figure 3, consists of 10 classes of machine printed 16 by 24 bit binary characters 'A' to 'K' (not including 'I') with 10 characters per class.

The complete data set is presented to the network during training until convergence, deemed to occur when the respective outputs for all given patterns changes by less than 0.1% over subsequent presentations of the data set. The code is taken to be the output of the node after convergence.

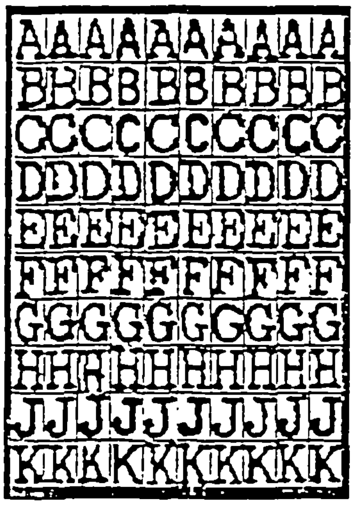


Figure 3. Data Set for Retrieval Experiment

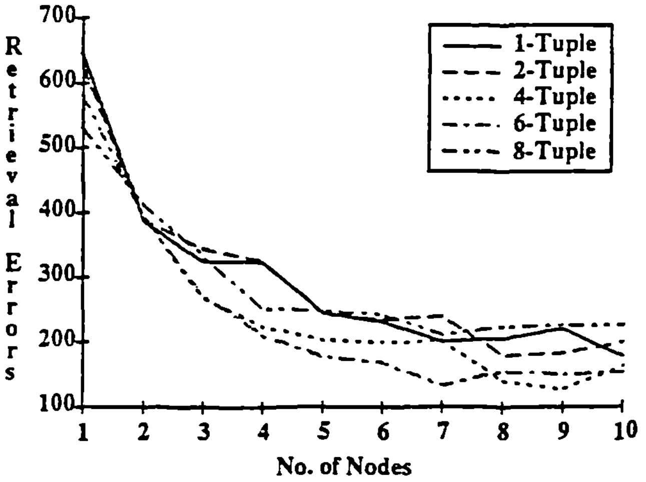


Figure 4. Variation of code performance with number of nodes for database shown in figure 3.

Images are matched against a target image, selected from the database, by retrieving those images whose codes are closest in the Euclidean sense to the target code. The number of images extracted from the database for each target corresponds to the number of examples in each class, in this case 10. The object of the retrieval mechanism is to retrieve all images from the same class as the target image and no other class. An error is defined as a class mismatch between the target and any one of the 10 retrieved images. The test is repeated for each and every image within the database as the target. The total number of errors is defined as the number of mismatches for the complete data set. Thus, the maximum total error for this data set is 100 x 10 = 1000.

The results are shown in figure 4. These show that a neural network trained using the learning algorithm presented here can be used to transfer the images into a low dimensional bound - the node behaves rather like an information filter. The information within a set of data from any bounded domain may be represented by a finite number of nodes - the more correlated the data, the fewer the number of nodes required. Successive nodes provide progressively less information, until the addition of additional nodes hardly furnishes us with any extra information at all. Figure 4 indicates that extending the system beyond 6 nodes reaps meagre rewards for the dataset in figure 3.

6 Discussion

Dimensionality reducing and classifier systems have contrasting requirements. In our coding scheme a 1-tuple best preserves the relative distances between the patterns in the original domain. This enables high dimensional images to be compared using low dimensional codes. As the node the tuple size is increased the node begins to behave more like a classifier and tends to cluster the training data. In such a system the distance preserving nature of the transform is degraded as the tuple size grows.

The size of the feature space in n-tuple systems increases exponentially with tuple size. This expanded feature space enables successive nodes to learn more orthogonal features. Thus, for multi-node systems, a large tuple size helps to ensure that successive nodes do not learn information conveyed by other nodes. This becomes more important as the number of nodes is increased. There comes a point where the advantages afforded by the expanded feature space for large tuples is offset by the distance preserving qualities of small tuples. The results shown in figure 4 bear this out.

A variant of this learning law may also be supported by conventional sum-of-weights type nodes [7].

7 References

1. Allinson. N. M., Johnson. M. J. and Moon. K. J.' *Digital Realisation of Self-Organising Maps.'* Neural Information Processing Systems Vol 1. pp. 728-738. Eds. Touretzky. D. S.,Morgan Kauffman Publishers, 1989.
2. Kerrin. M. A., Stonham. T. J. *'A Self-Organising strategy for Digital Neural Networks'.* Journal of Intelligent Systems. Vol. 2 Nos. 1-4., pp. 291-311, *1992.*
3. Linsker, R. *'Self-Organisation in a Perceptual Network'.* Computer. March *1988*, pp. 105-117.
4. Ntourntoufis. P. *'Self-Organising Properties of a discriminator based Neural Network'.* Proc. IJCNN, San-Diego, USA. Vol. 2. pp.319-324. *June 1990.*
5. Oja, E. *'A Simplified Neuron as a Principal Component Analyser'.* Journal of Mathematical Biology Vol.15. pp.267-273. *1982.*
6. Oja, E. *'Neural Networks, Principal Components and Subspaces'.* International Journal of Neural Systems. Vol. 1 pp.61-68. *1989.*
7. Rickman. R. and Stonham. T. J. ' *Similarity Retrieval from Image Databases - Neural Networks can Deliver.'* IS&T/SPIE International Symposium on Electronic Imaging: Science and Technology (Sub group: Storage and Retrieval for Image and Video Databases.). San Jose. USA. *Jan 1993.* (in print).
8. Rickman. R. and Stonham. T. J. *'A Self-Organising Logical Neural Network for Binary Data.'* Proc. Int. Joint Conf. on Neural Networks (IJCNN), Beijing, China, Vol. 2 pp. 745-752. *Nov. 1992.*
9. Sanger, T. ' *Optimal Unsupervised Learning in a Single Layer Linear Feedforward Network'.* Neural Networks. Vol. 2 pp.459-473. *1989a.*

Similarity retrieval from image databases - Neural Networks can deliver.

Rick Rickman and John Stonham

Neural Networks and Pattern Recognition Group
Department of Electrical and Electronic Engineering
Brunel University, Uxbridge, Middx UB8 3PH, U.K.

ABSTRACT

We address some of the problems of accessing database images which do not contain any indexing information and investigate methods of automating search strategies which currently rely on human operators to match the target against a number of images in the database. Such problems might include the extraction of facial photographs from a library given a suspect or the registration of new Trade Marks whose uniqueness must be assured. The object of the retrieval mechanism is to narrow down the search space for final perusal by the human operator.

We present a Neural Network based coding scheme to retrieve images from a database according to the degree of similarity with a target image. The code represents each image with respect to a set of feature archetypes learnt by the Neural Network during a training phase. We introduce a novel Neural Network learning law which performs an extremely efficient implementation of Principal Component Analysis and maximises the amount of information conveyed by the code. We present results using a database of machine printed fonts and discuss how the image size, the database diversity and code length affect the efficacy of the retrieval mechanism.

1.0 INTRODUCTION

Image retrieval and matching mechanisms are seen as an important aspect of database systems design and considerable attention has been focused on strategies which enable images to be extracted from a large data base according to a measure of similarity with a target [1-2]. We address some of the problems of accessing database images which do not contain any indexing information and investigate methods of automating search strategies which currently rely on human operators to match the target against a number of images in the database.

Any system which attempts to retrieve an image from a database where no indexing information is available must decompose each image into its characteristic set of features. Extracting features which characterise an image has proved to be an extremely difficult process to automate using conventional image processing techniques as those deemed to be important to a human operator are often not amenable to deterministic analysis. For this reason most schemes which attempt this kind of matching rely on human operators to analyse the image and select the feature codes manually from a library of predefined image entities. In this case the description of the image is dependent upon the extent of the code book. This is something of a drawback which can only be circumvented by limiting the diversity of images within the database or by continually updating the code-book. Despite efforts to standardise the description of images, such systems are often prone to subjective interpretation. In any event, manual coding of database images is invariably a time consuming and costly exercise.

. Neural Networks avoid the need for prescriptive descriptions of image artefacts by learning the salient features during a learning phase analogous to the type of learning found in biological processing systems. We propose a scheme which employs an array of Neural Networks to transform an image into a very succinct code which facilitates rapid fuzzy matching of images within the data base.

2.0 PRINCIPAL COMPONENT ANALYSIS FOR IMAGE CODING

Central to the thrust of any pattern recognition task is the notion of dimensionality reduction. The pattern domain typically contains large amounts of data which presents a formidable processing task if the features within the image are regarded as isolated entities. In meaningful real-world images however, there exist significant correlations between features of both a spatial nature and with respect to the contents of the database which allows the image to be represented in a much lower, and so more manageable, domain. Any image coding scheme should attempt to extract these correlations in order to capture the underlying trends within the data - the higher the degree of correlation the more succinct the code. The efficacy of a system which attempts to encode the images within the data base rests with its ability to extract these features.

The motivation behind the factors affecting the selection of such features has much in common with a classical image processing technique called Principal Component Analysis (PCA) [3]. The object of PCA is to extract the most important features, or Principal Components, from an ensemble of data so that a replica of any of the data items can be reconstructed from a linear recombination of these features in the appropriate proportions. Dimensionality reduction is achieved by virtue of the fact that most of the information within an image is contained within the first few Principal Components and relatively few are required for a faithful reconstruction of the image. PCA can be used as a coding mechanism by storing the proportion of the most significant principal components of that image as an element within the code - the greater the number of components used the more complete the representation of that image. The number of Principal Components required is dependent upon the diversity of the data set - a highly correlated data set requires very few Principal Components.

A similar technique, employing Karhunen-Loeve Transformations, has been used to encode human faces with limited success [4].

PCA requires that we calculate the Eigenvectors of the autocorrelation matrices of the data set. These calculations are extremely computationally expensive which renders such a method impractical for all but small sets of trivial images. However, Sanger [5],Oja [6,7] and Linsker[8] have shown that an approximation of PCA can be implemented very efficiently using a Neural Network topology if the size of the input domain is large and the number of Principal Components small. Such Networks can learn discriminant features automatically and provide a mechanism for self evolving coding topologies.

3.0 PRINCIPAL COMPONENT ANALYSIS WITH NEURAL NETWORKS

We propose a coding scheme which employs an array of Neural Networks to learn the Principal Components of the set of images within the database. This is shown below in fig.1. If the it has been trained successfully then presentation of an image containing a feature learnt by a Network will evoke a response from that node. The magnitude of that response indicates the extent of the feature present within the image. To a large extent the Principal Components will preserve the relative distance relationships between the images so that images whose codes are closest in the Euclidean sense are correspondingly close in the original pattern domain. This provides the basis of our matching metric.

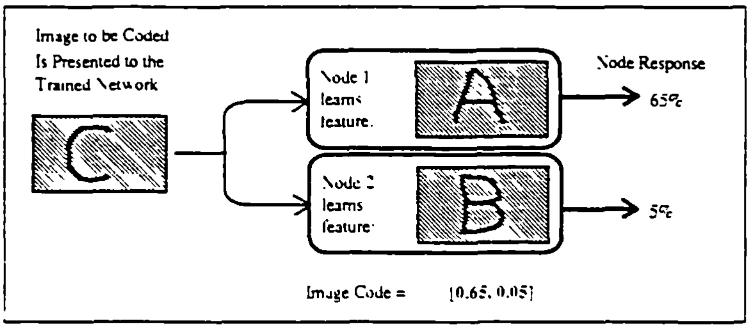


Fig.1. A Simple Neural Network Based Image Coding Scheme

The basic functionality of each node within this array of Neural Networks, shown in fig.2, is remarkably simple: the output of the node is just the weighted sum of its inputs. The weight determines the importance that an input has in defining the solution to a particular problem with respect to the complete data set. The weights are adapted iteratively in a training phase during which the entire data set is presented to the node. A learning law, found to exist in 'real-life' neurons, adapts the weights to correlate the input and output activity of the node. After convergence, which typically requires several passes through the data set, the node develops a transformation directly equivalent to, though only an approximation of, PCA[6]. The accuracy of this approximation is determined by the rate at which learning is undertaken within the system. If the learning rate is small, the approximation is accurate, the weights will converge to the Principal Components of the data set but will take a long time to get there. If the rate is high, convergence will be quicker but the approximation is less accurate and the resultant transformation may not be optimal [9]. The approximations inherent in the derivation of the learning law means that the learning rate can have a profound effect upon the performance of the code. The surest way to guarantee good performance is to set it very low. Unfortunately, this means that training times are very long. This is not a problem where the database remains fixed, so that the coding transformation remains constant, but is not practical where the transformation is optimised to adapt to the changing nature of the database.

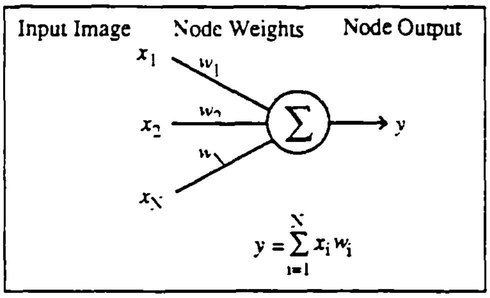


Fig.2. Functionality of a basic linear Neural Network.

Cottrell[10] presents a Neural Network architecture that will perform a coding transformation analogous to PCA but excessive training times render this method as impractical as a classical implementation of PCA. The learning law developed by Sanger[5] will learn faster than Cottrell's and has been used successfully as the basis of a coding scheme for an image database retrieval system[11]. Nevertheless, selecting an optimum learning rate for a particular application is typically a hit and miss affair requiring extensive trial and error testing before settling on a preferred value (indeed, this seems to be symptomatic of most Neural Network architectures). This state of affairs is unacceptable for a practical database system where the Neural Network must be trained *in situ* by an operator with only a modest level of expertise. In an attempt to alleviate this problem we have developed a completely novel Neural Network learning law which does not require any parameters to be set by the user yet converges extremely rapidly[9].

4.0 A NOVEL NEURAL NETWORK LEARNING LAW WITH NO PARAMETERS

4.1 Learning rule for a single node system

The basic structure and output function for the proposed Neural Network used in our coding scheme is identical to that shown in fig.2. The database is assumed to contain grey level images whose pixel luminosities have been normalised between '-1', for black, and '+1' for white. The complete training cycle for a single node system is outlined below:

i. Seed the weights of the node with an image selected randomly from within the database. Each weight monitors one pixel within the input image so that the number of weights corresponds to the number of pixels in the image. Note that the dimensions of all images must be normalised before presenting them to the network.

ii. Pass the entire contents of the database to the node and record the response of the node to each image.

iii. Reset all of the weights in the site to 0.

iv. Re-present each image and update the weights in the node according to the following learning law:

$$w_{i,t+1} = w_{i,t} - \lambda_u \left(\frac{y - \bar{y}}{K} \right) \qquad (1\cdot$$

Where:

$w_{i,t}$ is the value of the site before the update.

$w_{i,t+1}$ is the value of the site after the update.

x_{ij} is the value of the ith pixel in the jth image.

y_j is the response of the jth image obtained from stage ii.

$\bar{y}$ is the mean value of the node responses from stage ii. That is: $\bar{y} = \frac{1}{P}\sum_{j=1}^{P} y_j$ (2)

κ is a regulating term obtained from the node responses from stage ii. $\kappa = \sum_{j=1}^{P} |y_j|$ (3)

P is the number of images within the database.

v. Repeat stages ii to iv until the system converges.

This algorithm has been proved to converge to the Principal Components of the data set with probability 1[9]. Throughout training the dynamic range of the weights is automatically limited by κ so that $-1 \leq weights \leq 1$. A large weight indicates that a particular pixel within an image contributes a large amount of information to the code across the entire data set. Similarly, a pixel value close to zero indicates that the pixel contributes no information at all. It is instructive to see how a single node coding scheme trained using the learning law outlined above can produce a scalar valued code which preserves the relative distance relationships in the original pattern domain. Such a code will permit very rapid fuzzy matching of images from within the database.

4.2 Does the code preserve the relationships between the images ?

Consider the set of 16x24 bit binary characters, shown in fig.3 in which an 'A' font is gradually, and evenly, merged to a 'B' over 25 images. Image index '12' may be regarded as half way between an 'A' and a 'B. It is intuitively obvious that the data set could best be represented by a scalar from the node output if the pure 'A' and 'B' elicited maximal equal and opposite responses with image index '12' yielding a zero response. Since the images are merged linearly the code produced by the node output should also change linearly from image index '0' through to '25.

The node was trained using the learning algorithm outlined in 4.1 and converged after 4 passes, where convergence is deemed to occur when the respective node responses to each pattern differ by less than 0.1% of their former value. The results are shown below in fig. 3.

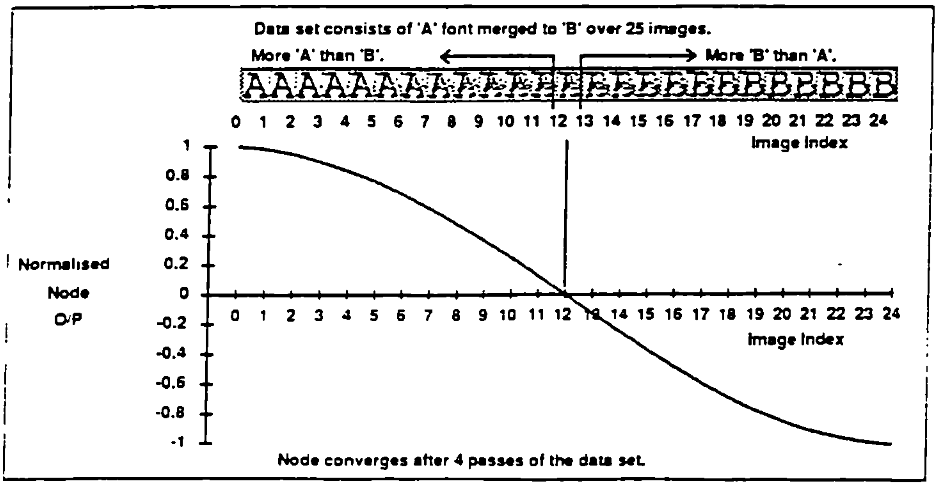


Fig.3. Test to show how the code preserves the relative distances between the images in the original pattern space.

The node behaves well as dimensionality reducer and its near linear response to successive images means that the code produced by the output could be used to match one image against another according to a measure of similarity. The resulting transformation is not completely linear however and exhibits slightly sinusoidal characteristics which does not quite preserve the exact distance relationships between the patterns in the original domain. This effect is most pronounced where the magnitude of the node output is large but the coding transformation is still adequate for our matching algorithm.

4.3 Learning rule for a multiple node system

It is obviously rather hopeful to expect the vast amount of information contained within an image database to be represented adequately by a scalar valued code. Thus, we must expand our scheme to a vector code, achieved by increasing the number of nodes within the system so that the output from each node forms an element of that vector, similar to the scheme outlined in fig.1.

Our learning paradigm produces a transformation which is directly equivalent to Principal Component Analysis and successive nodes should learn successive Principal Components of the data set. The first Principal Component, equivalent to the weight vector in a single node system, produces a linear transformation which maximises the variance of the node activity across the data set. The second Principal Component is constrained to be orthogonal to the first and the transformation again maximises the variance of the output of the second node subject to this constraint. The third Principal Components produces a similar transformation orthogonal to both the first and the second and so on. Generally speaking, successive Principal Components produce ever decreasing node variances and capture progressively less important artefacts within the data.

We can induce successive nodes to learn progressively orthogonal features by detuning the impetus for learning in subsequent nodes where the response of former nodes is large. This can be implemented by regulating the input to successive nodes as:

$$x_{ijk-1} = x_{ijk} - w_{ik}\left(\frac{y_{jk} - \bar{y}_k}{\varsigma_k}\right) \tag{4}$$

Where:

x_{ijk-1} is the value of the i_{th} pixel from the j_{th} image to the $k-1_{\text{th}}$ node in the system.

x_{ijk} is the value of the i_{th} pixel from the j_{th} image to the k_{th} node.

w_{ik} is the value of the i_{th} weight on the k_{th} node.

y_{jk} is the response of the k_{th} node to the j_{th} image.

$\bar{y}_k$ is the mean response of the kth node. That is: $\bar{y}_k = \frac{1}{P}\sum_{j=1}^{P} y_{jk}$

$$\varsigma_k = \sum_{i=1}^{N} w_{ik}^2 \tag{5}$$

N is the number of pixels in each image. The images must be normalised so that N is constant across the whole dataset.

The architecture to support this learning rule is shown in fig.4

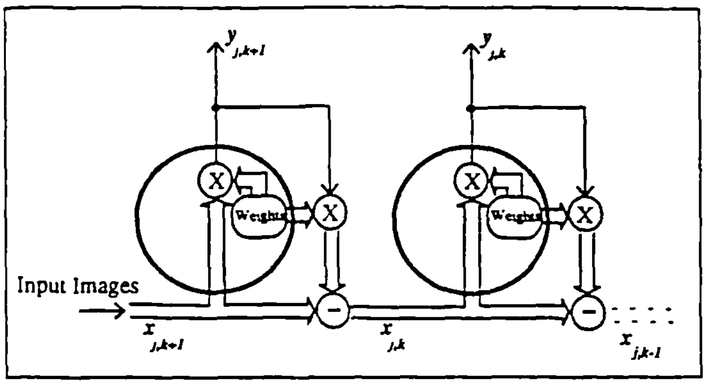


Fig.4. Functional Architecture to support learning in a multi-node system.

It is important to note here that the weight vector used in equation (4) is *not* the value of the weight vector during the iteration but the value *after* the previous complete training cycle outlined in steps ii. to iv of section 4.1. This means that one node must complete a pass through the training set in order to calculate the effect of orthogonalisation on the inputs to subsequent nodes. This may be implemented in one of two ways: a) by buffering the weight vectors from the previous update and using this value to calculate the orthogonalised input to the next node; or b) pass the complete data set to the first node, suspend learning on that node whilst passing the complete data set to the second node and so on. Method a) requires twice as much memory as method b) but will converge faster as the nodes learn concurrently. The speed up factor is most marked when the time to bring the image files into the system memory is significant. Our simulations adopt method a).

5.0 RESULTS

5.1 Data set used in simulations

The following test provides an objective measure of the performance of an image retrieval mechanism. The data set shown in fig.4, consists of 10 classes of machine printed 16 by 24 bit binary characters 'A' to 'K' (not including 'I') with 10 characters per class.

The complete data set is presented to the network during training until convergence, deemed to occur when the respective outputs for all given patterns changes by less than 0.1% over subsequent presentations of the data set. The code is taken to be the output of the node after convergence.

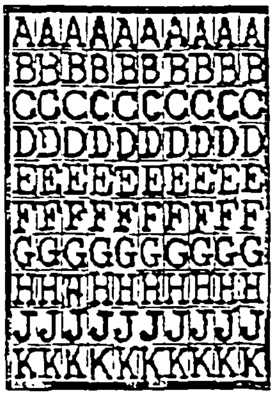


Fig. 5 Data Set for Retrieval Experiments

Images are matched against a target image, selected from the database, by retrieving those images whose codes are closest in the Euclidean sense to the target code. The number of images extracted from the database for each target corresponds to the number of examples in each class, in this case 10. The object of the retrieval mechanism is to retrieve all images from the same class as the target image and no other class. An error is defined as a class mismatch between the target and any one of the 10 retrieved images. The test is repeated for each and every image within the database. The total number of errors is defined as the number of mismatches for the complete data set. Thus, the maximum total error for this data set is 100 x 10 = 1000. In the following experiments the relative retrieval error allows a comparison between the performance of the retrieval mechanism for different sized databases and is defined as: (Actual Retrieval Errors/Maximum Possible Retrieval errors).

The data set used in these simulations is binary. However, the method is equally applicable to grey scale images.

5.2 Variation of Code performance with number of nodes.

The relationship between the standard deviation, code performance the number of nodes is shown in figs. 6 and 7 respectively. Successive nodes provide progressively less information until the addition of additional nodes hardly furnishes us with any extra information at all. Fig. 6 shows that extending the system beyond 6 nodes reaps meagre rewards.

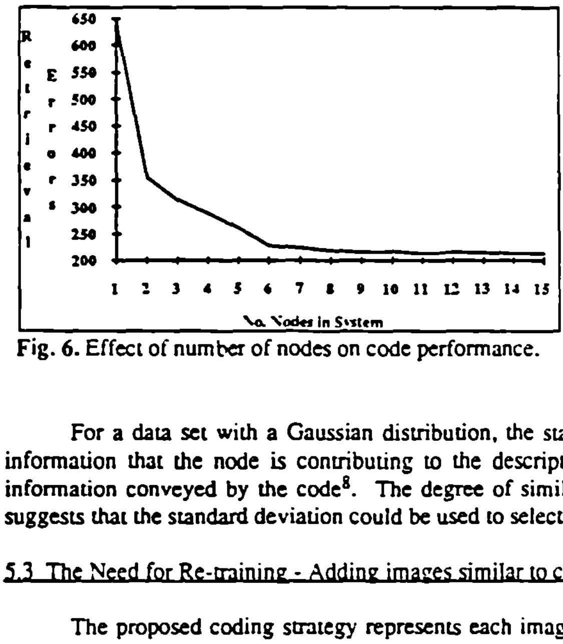


Fig. 6. Effect of number of nodes on code performance.

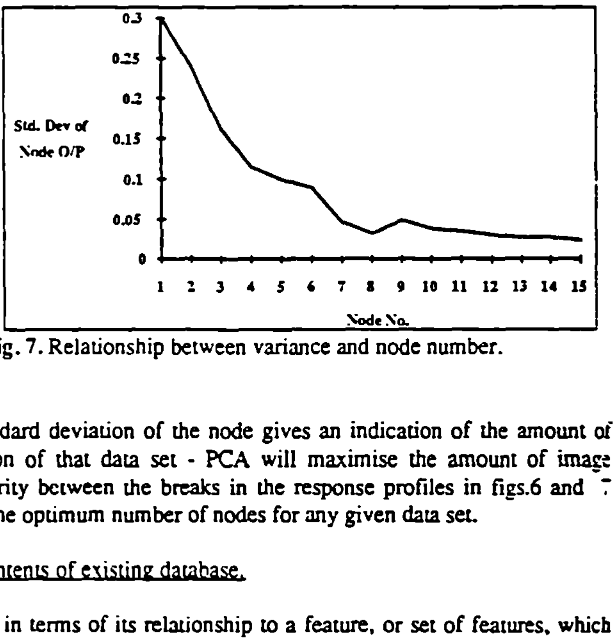


Fig. 7. Relationship between variance and node number.

For a data set with a Gaussian distribution, the standard deviation of the node gives an indication of the amount of information that the node is contributing to the description of that data set - PCA will maximise the amount of image information conveyed by the code[8]. The degree of similarity between the breaks in the response profiles in figs.6 and 7 suggests that the standard deviation could be used to select the optimum number of nodes for any given data set.

5.3 The Need for Re-training - Adding images similar to contents of existing database.

The proposed coding strategy represents each image in terms of its relationship to a feature, or set of features, which characterise the spread of images within a database. As images are added to or removed from the database it is likely that the distribution of patterns will change. This, in turn, will alter the nature of the features which permit optimal descriptions of the images therein. If the contents change significantly then the nodes will have to be re-trained to re-define the coding transform. The question we wish to address at this juncture is 'As the database changes, both in size and diversity, when does it become necessary to re-train the system ?'

In the following test a 6 node system node is trained with the data set shown in fig. 5 until convergence, deemed to occur when the difference between node responses on subsequent passes is less than 0.1%. The contents of the database are then changed but its diversity is kept fixed by keeping the number of classes of characters within the database constant. Note that the node is not re-trained; the new images are coded using the existing weights. The results are shown in fig.8. The training times required for re-training using the existing weights and 'starting from scratch' are also shown.

Data Set Size	Relative Errors (no re-training)	Relative Errors (with re-training)	Passes to converge (training from reset weights)	Passes to converge (training from existing weights)
10 Classes of 10	0.2150	-	42	-
10 Classes of 12	0.2361	0.2367	75	32
10 Classes of 14	0.2464	0.2408	43	29
10 Classes of 16	0.2387	0.2344	50	30
10 Classes of 18	0.2466	0.2391	49	25
10 Classes of 20	0.2460	0.2395	45	25

Fig. 8. Effect of re-training for expanding database of constant diversity.

These results show that the relative performance of the code remains remarkably constant as the database expands provided that the images that are added are similar to those images trained into the node. Indeed, if the new images are similar to the existing ones then the improvement in performance afforded by re-training is quite marginal. This is rather convenient as it does away with the need for 'on-line' training which is likely to be quite a time consuming process. A practical expedient would be to re-train the system during a 'down-time' at regular intervals.

5.4 The Need for Re-training - Adding images dissimilar to contents of existing database.

The experiment outlined above in 5.3 is repeated here but instead of adding images from classes already within the database we add images from completely new classes in blocks of 10 per class (in this case we gradually add characters 'L' to 'V' not including 'O' in blocks of 10). The results are shown below in fig. 9.

Data Set Size	Relative Errors (no re-training)	Relative Errors (with re-training)	Passes to converge (training from reset weights)	Passes to converge (training from existing weights)
12 Classes of 10	0.2508	0.2208	50	45
14 Classes of 10	0.3143	0.2679	74	46
16 Classes of 10	0.3406	0.29872	36	40
18 Classes of 10	0.3256	0.27892	67	60
20 Classes of 10	0.3270	0.2700	70	55

Fig. 9. Effect of re-training for expanding database of increasing diversity.

These results show, as we might expect, that as the database becomes more diverse the efficacy of the retrieval mechanism begins to drop off. Because the additional images are from different classes to those already contained within the database the existing weights are likely to produce sub-optimal codes. The results bear this out.

Under these conditions retraining will improve the performance of the retrieval mechanism more than the case mentioned in 5.3 but the performance without re-training is still quite good. This suggests that the node has learn a set of image primitives which can be used to describe a broad class of image types, even those not seen during training. The greater the diversity of the database, the more likely it is that the node will begin to learn such primitives. Under these conditions the effects of re-training become increasingly marginal.

5.4 The effect of image size on performance of system.

To observe the effect of the image size on training and code performance, the test outlined in 5.2 was repeated for a 6 node system with a scaled up versions of the same dataset. The characters were expanded to 32x48 bits and 64x96 bits respectively. The retrieval performance, the number of passes to convergence and the standard deviation of the node output are identical to the results shown in 5.2. This shows that the performance of the code is insensitive to the size of the images within the database.

6.0 CONCLUSIONS

The results presented in this paper indicate that a Neural Network can form the basis of a coding scheme to facilitate a fast and fuzzy retrieval mechanism for image databases. The Neural Network is shown to produce an extremely efficient implementation of PCA which preserves the maximum amount of information contained within an image.

Unlike most Neural Network paradigms, the learning law introduced here does not require user selection of any system parameters which is a distinct advantage where the system is to be used by operators with little, or no, Neural Network expertise.

Successive nodes learn features with decreasing significance. For a given data set there comes a point where the inclusion of additional nodes provides little or no additional information. Thus, it is possible to optimise the code length with respect to the contents of the data base. For a database of moderately diverse image types (so that the distribution tends to that of a Gaussian) the standard deviation can be used to select the optimal number of nodes which occurs where successive nodes exhibit standard deviations of the same order of magnitude.

As the diversity of the database increases, the nodes begin to learn generalised image primitives such as continuous line segments, for example. The upshot of this is that the need for re-training is less pressing as these primitives are pertinent to a very broad range of images.

One of the great strengths of our coding scheme lies in its ability to adapt to the changing nature of the database. As new images are added to the database there will come a time when the nodes need to be re-trained although the degradation in performance is gradual and, where the database contains a broad range of image types, almost imperceptible.

Linsker[8] has investigated a Neural Network learning paradigm which produces the same transformation as our own learning law, although the laws themselves are actually quite different. He showed that, when trained with a broad range of image types, the weights converge to feature primitives which characterise the general underlying characteristics common to *all* image types. This work is exciting in that these artefacts bear a remarkable resemblance to the so-called 'retinal fields' found to exist within mammalian visual systems. These fields, located within the primary visual cortex, are used to encode images for subsequent processing stages 'higher up' in the cortex. This seems to validate the approach of our own coding methodology.

7.0 ACKNOWLEDGEMENTS

This research was funded by the British Library Research and Development Department, contract MO921.

8.0 REFERENCES

1. Managing Image Data Bases. IEEE Computing Magazine pp. 3 - 62. Dec 1989.
2. Chang, Shi-Kuo *'Principles of pictorial information systems design '*. Prentice Hall International. 1987.
3. Oja, E. *'Subspace Methods for Pattern Recognition'*. Letchworth: Research Studies. *1983*.

4. Kirby, M. and Sirovich, L. *'Application of the Karhunen-Loeve Procedure for the Characterisation of Human Faces'* IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.12 No.1, Jan 1990.
5. Sanger, T. *' Optimal Unsupervised Learning in a Single Layer Linear Feedforward Network'.* Neural Networks. Vol. 2 pp.459-473. *1989a.*
6. Oja, E. *'A Simplified Neuron as a Principal Component Analyser'.* Journal of Mathematical Biology Vol.15. pp.267-273. *1982.*
7. Oja, E. *'Neural Networks, Principal Components and Subspaces'.* International Journal of Neural Systems. Vol. 1 pp.61-68. *1989.*
8. Linsker, R. *'Self-Organisation in a Perceptual Network'.* Computer. March *1988,* pp. 105-117.
9. Rickman, R. and Stonham, T. J. *'A Self Organising Logical Neural Network for Binary Data'.* Int. Joint Conf. on Neural Networks. Beijing. Nov. 1992.
10. Cottrell, G. W., Munro, P. and Zipser, D. *'Learning Internal Representations from Gray-Scale Images: An example of Extensional Programming'.* Ninth Annual Conference of the Cognitive Science Soc. Seattle. pp. 462-473. Hillsdale: Erlbaum. *1987.*
11. Rickman, R. and Stonham, T. J. *'Coding facial images for Database retrieval using a Self Organising Neural Network'.* IEE Colloquium on Macnine Storage and Recognition of Faces. Digest No. 1992/017. Jan. 1992.
12. Baldi, P. and Hornick, K. *'Neural Networks and Principal Component Analysis: Learning from Examples without Local Minima'.* Neural Networks Vol. 2. pp.53-58. *1989.*
13. Clark, R. J. *'Transform Coding of Images'* Academic Press. *1985.*
14. Bourland, H. and Kamp, Y. *'Auto-Association by Multi-Layer Perceptrons and Singular Valued Decomposition'.* Biological Cybernetics Vol. 59. pp 291-294. *1988.*

A Self Organising Logical Neural Network for Binary Data

R. M. Rickman
T. J. Stonham

Neural Networks and Pattern Recognition Research Group
Dept. Electronic Engineering
Brunel University
Uxbridge
Middx UB8 3PH
UK

Abstract *A novel Self Organising Neural Network architecture is introduced which is shown to extract the Principal Components from binary data sets with very short training times. The net performance is insensitive to order of presentation of the data set, does not require a learning rate to be set and will not settle into false minima. The system can be readily expanded to perform non-linear transformations of the data through an efficient implementation of sigma-pi type units.*

The network is based upon an array of RAM like memory elements and can be realised in hardware so that extremely rapid training times are a viable prospect.

1.0 Introduction

The feature extraction mechanism lies at the heart of any pattern recognition system. The original pattern domain is typically too large in which to undertake the recognition process and it is the task of the feature extraction phase to isolate the salient features which characterise the data set and represent them in a smaller and so more manageable domain.

Self Organising Neural Networks are capable of attuning to important features within a data set and can automatically develop a transformation, through a training phase, which maps the original pattern into a very compact representation so that subsequent pattern matching may be carried out with comparative ease.

Sanger [1], Oja[2], Linsker[3] and others have demonstrated how a simple linear Neural Network can be used to extract the Principal Components of a data ensemble. The Principal Components represent an optimal linear co-ordinate transformation which maximises the variance of the node response to the training data and so accentuates its discriminant features. Principal Components correspond to the eigenvectors of the autocorrelation matrix of the data and dimensionality reduction is achieved by virtue of the fact that relatively few Principal Components are required to represent each pattern within the set. Sangers work shows that a Self Organising Neural Network Node can perform a very efficient approximation to Principal Component Analysis (PCA) where the input patterns are large and the number of PC's small.

One of the great shortcomings of the sum-of-wieghts type PCA nodes, and indeed most Neural Network paradigms, is that they are sensitive to the various network parameters which must be carefully honed to suit the data set. To a certain extent it is possible to circumvent these problems by opting for very low training rates which incurs very long training times.

An entirely novel Self Organising Neural Network architecture based on RAM like memory elements is presented which makes no assumptions regarding the underlying statistical nature of the data (except, of course, that there are features to extract) and can be trained extremely rapidly to encode binary data sets.

2.0 A Logical Solution to Self Organisation

The proposed Self Organising architecture, shown below in fig. 2.0, consists of an array of real valued memory elements which are addressed by the binary pattern on an input retina. Location 0 to each memory is addressed by a 'white' element on the retina and location 1 by a 'black'. The output of the node is simply the sum of the sites addressed by the pattern.

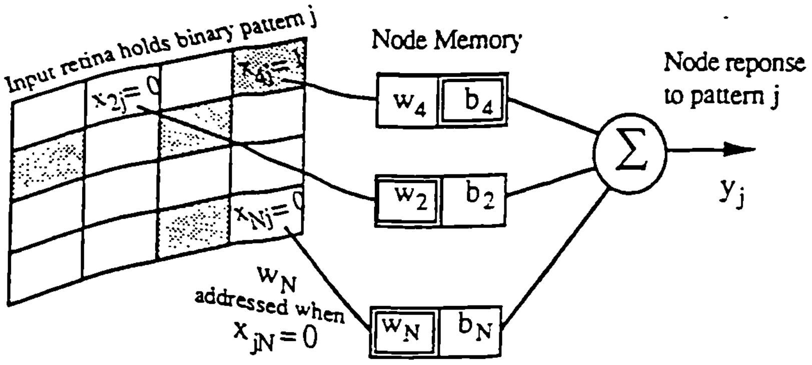

Input retina holds binary pattern j
$x_{2j}=0$
$x_{4j}=1$
$x_{Nj}=0$
w_N addressed when $x_{jN}=0$
Node Memory
w_4
b_4
w_2
b_2
w_N
b_N
Σ
Node reponse to pattern j
y_j

Fig. 2.0 Basic architecture for the Self Organising Logical Neuron.

The objective of the training is to fill the sites so that the variance of the node response to the complete data set is maximised. This may be achieved by updating the sites commensurately with the strength of the node response. However, such a learning law will tend to accentuate those areas within the pattern which have a large DC component and thus contain no discriminant information. This can be circumvented by forcing the mean of the node response to zero so that the total update for a pair of sites addressed by every pattern in the data set is zero. Training the network consists of the following phases:

i. Clear the network by writing a '0' to every memory location.
ii. Seed the node by writing a '1' to those sites addressed by a pattern chosen arbitrarily from the data set.
iii. Present each pattern to the node and store its response. Repeat for all patterns.
iv. Clear down the network as in stage i.
v. Re-present each pattern to the node and write the normalised, mean-corrected response to the sites addressed by that pattern.
vi. Repeat stages ii to vi until convergence.

3.0 Analysis of the Learning Rule

After initialisation the response of the node to pattern j is given by:

$$y_j = \sum_{i=1}^{N} b_i x_{ij} - \sum_{i=1}^{N} w_i(1-x_{ij}) \qquad (1$$

Where:

x_{ij} is the value of the binary element i in the jth pattern.

$x_{ij} = 1$ for a 'black' element (binary 1).

$x_{ij} = 0$ for a 'white' element (binary 0).

b_i is the value stored at the memory addressed by the ith element when it is 'black'.

wi is the value stored at the memory addressed by the ith element when it is 'white'.

N is the dimensionality of the input pattern.

After one pass through the complete data set the new value of the i_{th} site is:

$$\widehat{b_k} = \gamma \sum_{j=1}^{P} (y_j - \bar{y})\, x_{kj} \qquad (2$$

$$\widehat{w_k} = \gamma \sum_{j=1}^{P} (y_j - \bar{y})(1 - x_{kj}) \qquad (3$$

Where:

γ is a constant to limit the dynamic range of the site values.

$\bar{y}$ is the mean response of the node for all P training patterns.

$$\bar{y} = \frac{1}{P} \sum_{j=1}^{P} y_j \qquad (4$$

For the j_{th} pattern $(y_j - \bar{y})$ will be added to either the b_i or w_i sites and after a complete training cycle:

$$\widehat{w_i} = -\widehat{b_i} \qquad (5$$

Then from (5 and (1

$$y_j = \sum_{i=1}^{N} b_i\, (2x_{ij} - 1) \qquad (6$$

Combining (6 and (4

$$\bar{y} = \frac{1}{P} \sum_{i=1}^{N} b_i \sum_{j=1}^{P} (2x_{ij} - 1) \qquad (7$$

so that:

$$y_j - \bar{y} = 2 \sum_{i=1}^{N} b_i \left(x_{ij} - m_i \right) \qquad (8$$

Where:

m_i is the mean number of 'black' elements in position i across the training set.

$$m_i = \frac{1}{P} \sum_{j=1}^{P} x_{ij} \qquad (9$$

Substituting (8 into (2 gives the new value of the site after one pass through the training set:

$$\widehat{b_k} = 2\gamma \sum_{i=1}^{N} b_i \sum_{j=1}^{P} x_{kj}\, (x_{ij} - m_i) \qquad (10$$

In matrix notation the state equation for the learning rule becomes:

$$\widehat{b_k} = 2\gamma \left[b_1 \, .. \, b_N \right] \left[x_{k1} \, .. \, x_{kP} \right] \begin{bmatrix} x_{11}-m_1 & .. & x_{1P}-m_1 \\ : & & : \\ x_{N1}-m_N & .. & x_{NP}-m_N \end{bmatrix}^T \qquad (11$$

Since x_{ij} is binary valued:

$$\widehat{b_k} = 2\gamma \left[b_1 \, .. \, b_N \right] \left[x_{k1}-m_k \quad .. \quad x_{kP}-m_k \right] \begin{bmatrix} x_{11}-m_1 & .. & x_{1P}-m_1 \\ : & & : \\ x_{N1}-m_N & .. & x_{NP}-m_N \end{bmatrix}^T \qquad (12$$

Which can be written in vector form as:

$$\widehat{B} = 2\gamma\, B\, Q \qquad (13$$

Where:

$\widehat{\mathbf{B}}$ is the new site values $\left[\widehat{b_1} \,..\, \widehat{b_N}\right]$ after one training cycle.

B is the old site values $\left[b_1 \,..\, b_N\right]$ before training.

Q is the autocorrelation matrix of the data set:

$$\begin{bmatrix} x_{11}\text{-}m_1 & .. & x_{1P}\text{-}m_1 \\ : & & : \\ x_{N1}\text{-}m_N & .. & x_{NP}\text{-}m_N \end{bmatrix} \begin{bmatrix} x_{11}\text{-}m_1 & .. & x_{1P}\text{-}m_1 \\ : & & : \\ x_{N1}\text{-}m_N & .. & x_{NP}\text{-}m_N \end{bmatrix}^T$$

Because the 'black' and 'white' elements are mutually exclusive to each other the Q matrix holds for the 'white' site update too (see equation (3).

$$\widehat{W} = 2\gamma \, W \, Q \qquad (14$$

From (5:

$$\widehat{W} = -2\gamma \, B \, Q \qquad (15$$

Setting $\gamma = \frac{1}{2}\sum_{j=1}^{P} |y_j|$ will limit the sites to $-1 \leq \widehat{b}_i \leq 1$

Learning rules with state equations of the type shown in equation (14 have been thoroughly investigated and proven to converge to the eigenvectors of the autocorrelation matrix of the data set [1][2][3][4]. This learning rule is directly equivalent to the classical statistical technique Principal Component Analysis (PCA) but requires neither the direct computation of the autocorrelation matrix nor the eigenvectors which would otherwise be impractical for high dimensional data sets.

Additionally, unlike existing sum-of-weights type neural network implementations of PCA, no implicit assumptions are made regarding the statistical nature of the data set or approximations made in the derivation of the state equation for the learning rule. The upshot of this is that the learning rule does not require a learning rate to be set which in turn leads to faster and more accurate convergence.

Most neural network paradigms are sensitive to the order of presentation of the input data and are prone to settle into false minima (especially if the learning rate is set too high). Because this learning rule does not implement the system updates until all of the data has been presented to the net it does not suffer from this drawback and consequently will not converge to localised solutions.

It can be seen that equations (14 and (15 are effectively equivalent to each other and as such the update of the 'white' sites is inexorably bound to the update of the 'black'. One could argue then that the memory requirements of this system could be halved simply by considering update on black elements only. Whilst this is undoubtedly true, the notion that the 'white' and 'black' elements address distinct memory locations allows the learning law to be developed further to accommodate non-linear transformations.

4.0 A non-linear implementation of the learning rule

The learning rule outlined in (14 will perform a linear transformation of the input data to maximise the variance of the node response across the data set. The resulting conformal transformation will preserve the relative relationships between the patterns in the input and output domains. A coding scheme based on this transformation enables images to be compared in a very low dimensional bound and provides the basis of a very fast and fuzzy retrieval mechanism for large databases of non-indexed images[5].

However, a linear learning law will not extract any of the higher order features from the data set which are required for 'clustering' data, for instance. The above learning law can be modified to perform highly non-linear transformations using a technique developed by Aleksander *et al* [6].

The learning law outlined earlier maps one binary pixel to one site memory, so that a 'black' element will address one part of the memoiy and a 'white' the other. Non-linearity can be introduced into the model b collecting these address lines into fixed n sized bunches or 'n-tuples'. These n-tuples address a 2^n location memory which stores a site value in the same fashion described earlier .

Thus, a data set of 16 bit patterns would require 8 x 2-tuples (each tuple addresses a 4 location memory) or 4 x 4-tuples (each tuple addresses 16 location memory) and so on. Each address line within the tuple is mapped randomly onto the pattern and as in the earlier 1-tuple system the output of the node is simply the sum of the sites dressed by each tuple. It is easy to show that such a mapping will perform a transformation similar to the so-called sigma-pi nodes [7] and can elicit non-linear statistics from the input data - the order of the non-linearity increases with tuple size and it is possible to exercise considerable control of the nature of the features extracted by the node.

These non-linearities render the learning law given in (14 somewhat incomplete but the 'essence' of the system dynamics are still the same.

5.0 Testing the performance of a 1 node system

The object of this test is to show how a single node of the type discussed can condense an ensemble of high dimensional data to a single, scalar valued code. The tests permit a comparison between the performance of a linear , 1-tuple node and the sum-of-weights type PCA node outlined in [1]. The test also aims to show how the tuple size affects the linearity of the mapping and highlights the difference between conformal and 'clustering' type transformations in Self Organising systems.

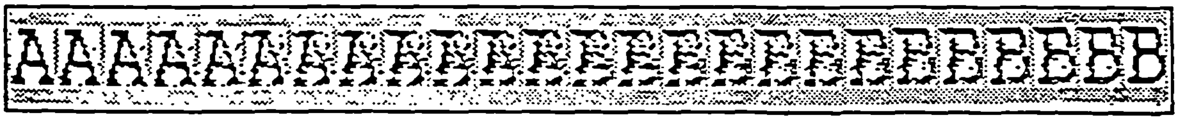


Fig 5.1 Data set used to test the performance of the Self Organising Nodes

The data set, shown in fig. 5.1 consists of a 16x24 bit binary 'A' character gradually (and evenly) merged to a 'B' character over 25 images. Because the change of the 'A' to the 'B' is linear across the data set we would expect an idealised conformal mapping to produce the code shown in fig. 6.3.

6.0 Results for a Self Organising Logical Node

The node was initialised by filling those sites addressed by a pattern, chosen arbitrarily from the data set, with a random seed value where $-1 \leq seed \leq 1$. The complete data set is presented to the network and the node response to each pattern stored (see equation (1.). After all of the patterns have been presented, the sites are cleared to zeros and the patterns presented again. The stored response from each pattern is normalised and added to the site value addressed by the appropriate input pixel (see equation (2.). This procedure is repeated until convergence - deemed to occur when:

$$\sum_{j=i}^{P} \Delta(s_j) < 0.0001$$

Fig. 6.1 shows the performance of the network for different tuple sizes. Fig. 6.3 shows the performance of a sum-of-weights type PCA node (outlined in [1]) for the same training set and indicates how the training rate for such nodes must be carefully chosen to avoid convergence to non-optimal solutions.

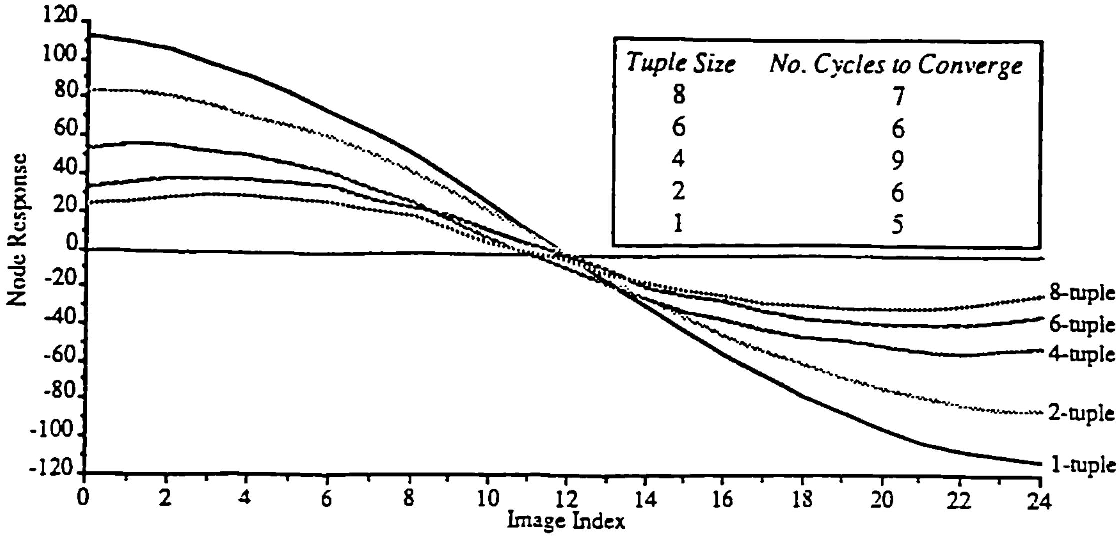


Fig 6.1 Response of Self Organising Logical Node to 'A' (index 0) to 'B' (index 24) data set.

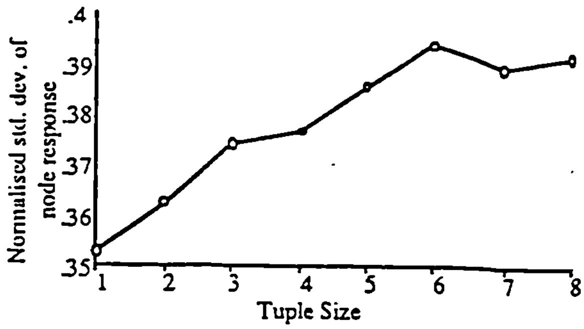


Fig. 6.2 Effect of tuple size on normalised standard deviation of Self Organising Logical Node.

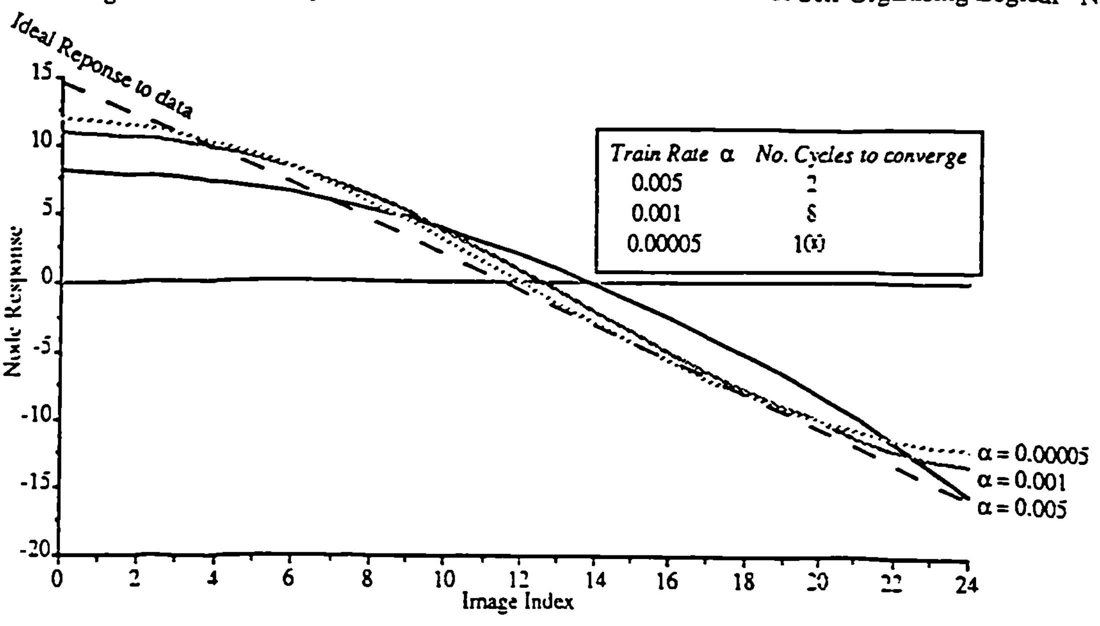


Fig. 6.3 Effect of training rate on performance of sum-of-weights type PCA Node.

7.0 Conclusions

This work shows that the simple architecture and training algorithm outlined in section 2.0 can perform very efficient calculation of the Principal Components of a binary data set. Unlike the more conventior sum-of-wieghts type PCA nodes this scheme does not require any training parameters to be set so that the netwo can be trained very rapidly without the danger of settling into so-called 'false-minima' (shown in fig. 6.3). The false-minima occur because the node tends to extract localised pattern relationships which are dependent upon tl order of presentation of the training data. In the proposed model the sites are not updated until all of the training da has been presented and this problem is alleviated.

A basic 1-tuple system will perform a linear transformation of the input to the output domain. Such conformal mapping is required for pattern coding systems, for instance. The system can be made to operate in non-linear fashion by increasing the tuple size - a larger tuple size will extract increasingly higher order statics fro the data set. In this case the node will tend to cluster the data set into distinct regions within the output doma which is useful for pattern classification tasks.

The system can be realised readily in hardware using RAM-type memory structures which offers the prospe of very rapid training times.

References

[1] Sanger T. D. (1989) *'Optimal unsupervised learning in a single layer linear feed forward Neural Network '*. Neural Networks Vol. 2, pp 459 - 473.

[2] Linsker R. (1988) 'Self Organisation in a perceptual Network'. Computer, 21. pp105-117.

[3] Oja E. (1982) *'A simplified neuron model as a Princial Component Analyser '* Journal of Mathematics and Biology, 15. pp 267 - 273.

[4] Oja E. (1989) *'Neural Networks, Principal Components and Subspaces'* .Int. Journ. Neural Systems Vol.1 No.1 pp.61 - 68.

[5] R. Rickman, T. J. Stonham (1992) *'Coding Images for Data Base Retrieval using Neural Networks'*. IEE Colloquim on. "Matching Storage and Recognition of faces ".London 1992.

[6] Aleksander I.,Stonham T. J. (1979) *'A guide to pattern recognition using Random Access Memories '*. IEE Proc. on Comp. and Digital Techniques 1979 Vol.2, pp 29 - 36.

[7] Gurney K. N. (1992) *'Training Nets of Hardware Realizable Sigma-pi Units'* Neural Networks 5 pp 289 - 303.

CODING FACIAL IMAGES FOR DATABASE RETRIEVAL USING A SELF ORGANISING NEURAL NETWORK

R. M. Rickman and T. J. Stonham

We address some of the problems of accessing database images which do not contain any indexing information. More specifically, we investigate methods of automating search strategies for facial images which currently rely on human operators to match the target against a number of images in the database. Such problems might include the extraction of facial photographs from a library given a suspect. The object of the retrieval mechanism is to narrow down the search space for final perusal by the human operator. This work shows how an automatic feature extraction system based on a Self Organising Neural Network topology can provide the foundations for a facial image database retrieval system.

Thrust of the Coding Scheme

The objective of the this research is to develop a self-evolving coding scheme which codes each image with respect to a set of feature archetypes which characterise the spread of images across the database. To extract the most salient database archetypes each image must be decomposed into a set of characteristic features. This decomposition process is a principal concern in the design of any non-indexed image retrieval system and poses some difficult problems as such features are rarely amenable to deterministic analysis [1]. Neural Networks avoid the need for prescriptive descriptions of image artifacts by learning the salient features during a training phase [2].

We propose a coding scheme which employs an array of Neural Networks which can .learn important features across the database [3][4]. Presenting an image to a trained Neural Network will evoke a response if the input image contains features similar to those learnt by that node. The magnitude of the response indicates the extent of the feature present within the image. Thus, a simple coding scheme could be designed by using the output of each Net to form an element of a similarity vector. Images with close similarity vectors could then be deemed to be similar. Each image in the data base is coded with respect to a set of salient features which characterise the diversity of its contents. Such a scheme is shown in fig. 1.

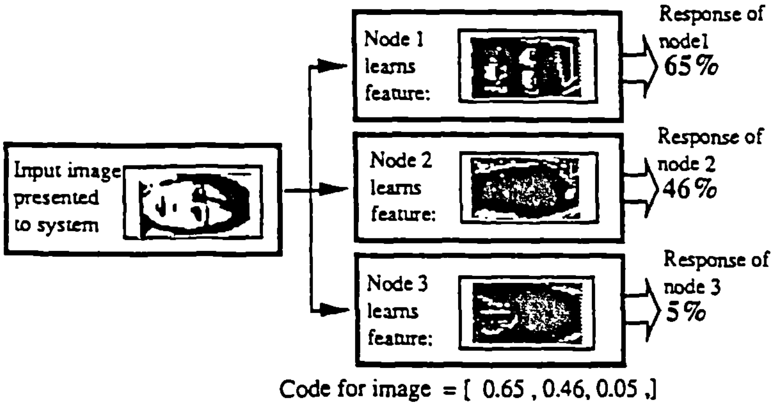


Fig. 1 A simple Neural Network Image Coding Scheme

Rick Rickman and John Stonham are part of the Neural Networks and Pattern Recognition Group Department of Electrical and Electronic Engineering, Brunel University, Uxbridge, Middlesex.

This coding process transforms the image into a succinct code which enables images to be compared in a very low dimensional bound with the degree of fuzziness approaching that of a human operator.

Each node within the Network is trained on orthogonal features which maximises the variance of the output. Such a feature enables greatest number of images to be encoded with optimum selectivity.

Principal Components and Neural Networks

The motivation behind the factors affecting the choice of features learnt by the Neural Network has much in common with a classical image processing technique called Principal Component Analysis (PCA) [5]. The object of PCA is to extract the most important features, or Principal Components, from an ensemble of data so that a replica of any of the data items can be reconstructed from a linear combination of these features. Each image can be represented by the proportion of the Principal Components. Relatively few Principal Components are required for an accurate reconstruction of the image - the number of Principal Components required is dependent upon the diversity of the data set.

An approximation of PCA can be implemented very efficiently using a Neural Network topology if the size of the input domain is large and the number of Principal Components small [6] [7]. Such Networks can learn discriminant features automatically and provide a mechanism for self evolving coding topologies.

For a linear Neural Network node the output is the weighted sum of the input vector:

$$y = \sum_{i=1}^{D} x_i \cdot c_i$$

Where y is the output of the node, x_i is the i_{th} input and c_i the i_{th} weight for a D dimensional image.

To induce Self Organisation the node learns input features which have a high correlation to the weight vector (ie when the output is large) - this is known as Hebbian Learning. Thus:

$$\Delta c_i = \alpha . x_i . y$$

Where Δc_i is the incremental change on the i_{th} weight and α is the learning rate (<<1).

As learning proceeds, the weight vectors begin to rotate in the direction of the Principal Components. To prevent saturation during training the learning resource is constrained so that:

$$\sum_{i=1}^{D} c_i^2 = 1$$

This yields the Self Organising learning law:

$$\Delta c_i = \alpha . x_i . y - y^2 . c_i$$

For a single node the weights converge to the first Principal Component of the input data. To induce orthogonal learning across the nodes, those features learnt by previous nodes are discounted from the learning process in subsequent ones. If the inputs do not have zero mean then the node tend to learn the mean components of the input ensemble rather than the discriminatory features. To correct for this, the mean of each input line to the node must be subtracted from the input prior to presentation to the network.

Coding and Matching Procedure

The coding phase may be summarised as follows:

1. Train the Neural Network to extract the Principal Components from the image data base. During training images are extracted from the data base and presented to the Neural Network. Training continues until the networks converge to the principal components of the system.

2. Each image within the database is presented to the trained Network and the node output recorded as an element of the discriminant code. The number of elements of the discriminant code corresponds to the number of nodes used in the system.

The objective of the retrieval mechanism is to narrow the search space for the operator down to a manageable size and to retrieve those images whose codes are closest to the input . The retrieval mechanism may be summarised as follows:

3. Encode the input image as in 2.

4. Scan all image codes within the database. Retrieve those images with the minimum least squares difference between corresponding code elements.

Results

The test database contained 500 32x59 binarised facial images of male Caucasians. The input test images consisted of an image from within the database corrupted both with noise and meaningful image artifacts (i.e. a change in hair style and facial features painted onto the original image). The objective of the retrieval mechanism was to extract the original uncorrupted image from the data base - ten images with the closest match are displayed. The system consisted of 8 nodes and output from each quantised to 8 bits and assigned to one element of the match code (ie each image is represented by a 64 bit code).

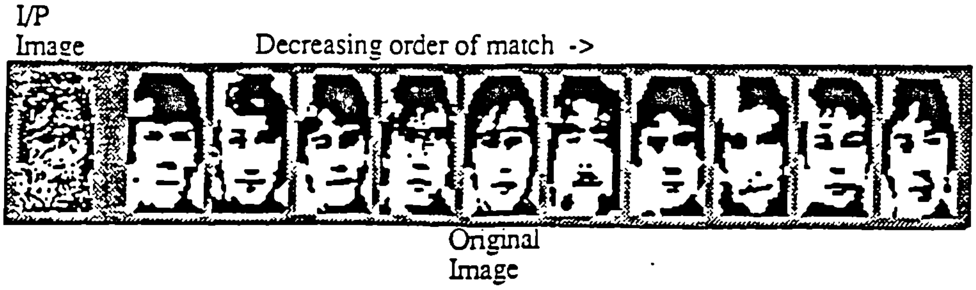


Fig. 2
Target Corrupted with:
25% Noise

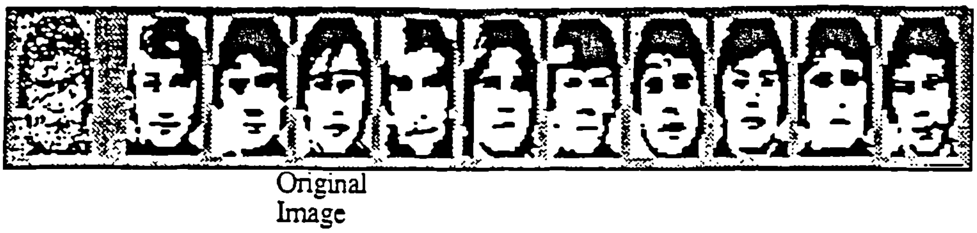


Fig. 3
Target Corrupted with:
15% Noise
+ 10% Artifacts

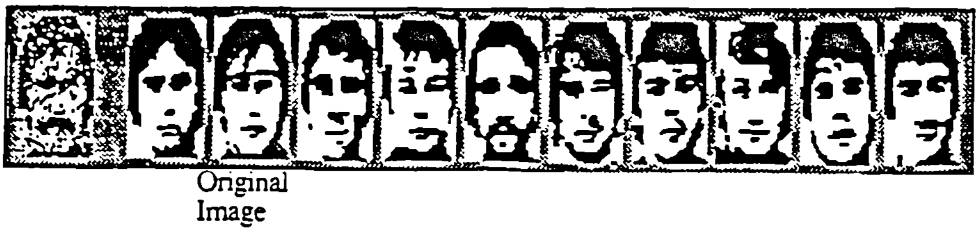


Fig. 4
Target Corrupted with:
15% Noise
+ 10% Artifacts

Conclusions

Results show that the sum-of-weights PCA Neural Network nodes can be used very effectively to fin the optimal discriminant features within low dimensional images. These features form the basis of coding scheme which can perform rapid fuzzy matching of images within a database.

References

[1] *Managing Image Data Bases.* IEEE Computing Magazine December 1989 pp. 3 - 62.

[2] R. Lipman 'Pattern Classification Using Neural Networks'. IEEE Comms. Magasine Nov. 89 pp 47 - 63.

[3] R. Rickman, T. J. Stonham *'Coding Images for Data Base Retrieval using Neural Networks'.* 3rd NERVES Neural Network Workshop Jan '91, Grenoble, France.

[4] R. Rickman, T. J. Stonham *'Images for Data Base Retrieval using Neural Networks''.* Research report
Nov '91. Dept. Elec. Eng., Brunel University, Uxbridge, Middx.

[5] M. Kirby, L. Sirovich *'Application of the Karhunen-Loeve Procedure for the Characterisation of Human Faces'.* IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.12 No.1, Jan '90.

[6] T. D. Sanger *' Optimal unsupervised learning in a single layer linear feedforward Neural Network '.* Neural Networks Vol. 2, pp 459 - 473.

[7] E. Oja *'Neural Networks, Principal Components and Subspaces'* .Int. Journ. Neural Systems Vol.1 No.1 pp.61 - 68.

CODING IMAGES FOR DATA BASE RETRIEVAL USING NEURAL NETWORKS

R. M. Rickman T. J. Stonham

Dept. Electronic Engineering
Brunel University
Uxbridge
Middlesex UB8 3PH
UK

Abstract. *We address some of the problems associated with systems which retrieve images from large data bases according to a measure of similarity with a target image. The efficacy of these systems rests with the ability of the scheme to extract and encode those features which characterise the image.*

Neural Networks offer great potential in this area as they can 'learn' the salient features within the image automatically. We show how an array of Neural Networks may be used to encode an image to facilitate a very rapid fuzzy match of images within the data base and discuss the factors affecting training of these networks. The coding scheme proposed employs a digital Neural Network architecture which utilises standard RAM technology so that a practical system is a viable prospect. Preliminary results are presented and propects discussed.

Introduction

Image retrieval and matching mechanisms are seen as an important aspect of data base systems design and considerable attention has been focussed on strategies which enable images to be extracted from a large data base according to a measure of similarity with a target image [1] [2]. Two potential applications of such a system are the extraction of facial photographs from a criminal library given a target image and the registering of new trade marks whose uniqueness must be assured.

Comparing the 'raw' target image with each and every image within the data base is computationally expensive and impractical where the data base contains many images. The searching mechanism can be made considerably faster by representing the salient features within each image as a code so that a match can be made by comparing the extent of shared features represented by the codes. Images whose code have a high degree of correlation with the target image can be extracted from the data base for perusal by the human operator. The objective of the system proposed here is to narrow down the search space for the operator to a manageable size.

The efficacy of any system which attempts to encode the images within the data base rests with its ability to extract the salient features and represent them as a consistent code which permits fuzzy matching of images with a degree of intolerance to position, scale lighting and other real world artifacts.

Extracting those features which characterise an image has proved to be an extremely difficult process to automate using conventional image processing techniques as those features deemed to be important to a human operator are often not amenable to deterministic analysis and defy prescriptive programming methodologies. For this reason most schemes which attempt this kind of matching rely on human operators to analyse the image and select the feature codes from a library of predefined image entities. The description of the image is dependent upon the extent of the code book and the coding process can be a time consuming and costly exercise.

Neural Networks have received much attention as processing architectures which learn to extract the characteristics of a data set presented during training. These architectures offer self-evolving coding capabilities which can give a constant response to a numerically indeterminate data set. We propose a scheme which employs an array of Neural Networks to transform an image into a very succinct code which facilitates rapid fuzzy matching of images within the data base.

Neural Networks as Dimensionality Reducers

Central to the thrust of any pattern recognition task, whether by a Neural Network or through more conventional techniques is the notion of dimensionality reduction. The pattern domain typically contains large amounts of data which presents a formidable processing task if the features within the image are regarded as isolated entities. In meaningful real-world images however, there exist significant correlations between features of both a spatial nature and with respect to the data set which allows the image to be represented in a much lower dimensional space. This reduced representation allows subsequent recognition phases to be undertaken in a manageable domain. The image coding scheme should attempt to extract these correlations within the pattern space and capture the underlying trends within the data - the higher the degree of correlation the more succinct the code.

Neural Networks are successful as pattern recognition architectures because the nodes which sample the input domain interact in a way which extracts the inherent correlations that exist within the data base. Hebbian learning, for instance, increase the activity the activity between nodes which sample coincidental features and decrease the activity between nodes which sample disjoint features. In this way higher order relationships between several entities within the image can be represented in the activity of just a single node in a completely non-deterministic manner. The representation of an image in a lower dimensional bound may be regarded as a Neural Network Transformation whose parameters are dependent upon the data set and where the transformation is learnt.

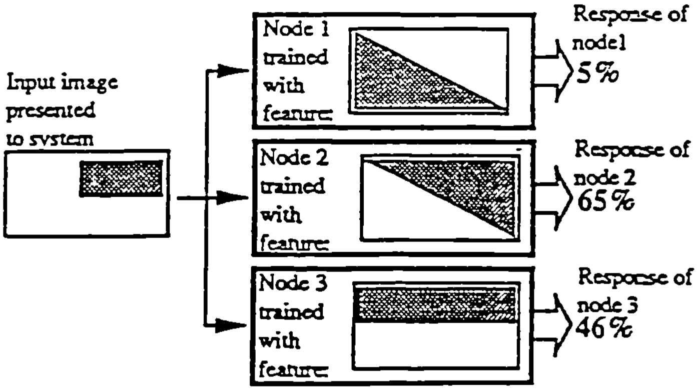

Input image presented to system
Node 1 trained with feature
Node 2 trained with feature
Node 3 trained with feature
Response of node1
5%
Response of node 2
65%
Response of node 3
46%
Code for image = [0.05 , 0.65 , 0.46]

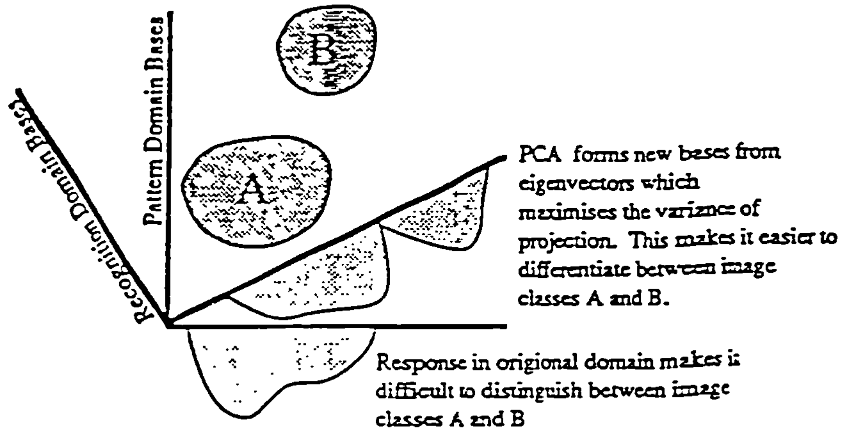
Fig. 1 A simple Neural Network Image Coding Scheme

The coding scheme proposed here, outlined in Fig.1, consists of an array of Neural Networks where the output from each net forms an element of the code. The network is trained on images from within the data base so that each builds an individual generalised internal representation of a feature prototype which acts as a discriminant function that reflects the spread of characteristics across the data base. This discriminant function may be considered optimal if the variance of the node response is maximised across the data set and is statistically independent from the response of the other nodes. The objective of the training is to form a diverse set of prototypes which represent the principal feature components with minimal overlap in pattern space.

Coding Images using Principal Component Analysis

Principal Component Analysis (PCA) is a well established technique which attempts to extract those features from an ensemble which most accurately reflect the characteristics of the data set [3].

This technique forms a set of orthogonal basis vectors (one for each Principal Component) on which the projection of the pattern space forms optimal discriminant functions with respect to the data set. PCA isolates the most significant eigenvector of the co-variance matrix of the training set. Any image within the data base can be reconstructed through a linear recombination of these eigenvectors in the appropriate proportions. Dimensionality reduction is achieved by virtue of the fact that most of the information within an image is contained within the first few eigenvectors and relatively few components are required for a faithful reconstruction of the image. The most significant principal components are those which provide optimal discriminant information and thus have the highest variance across the data set. This is shown in fig. 2. PCA can be used as a coding mechanism by storing the proportion of the most significant principal components of that image as an element within the code - the greater the number of components used the more complete the representation of that image.



PCA forms new beses from eigenvectors which maximises the variance of projection. This makes it easier to differentiate between image classes A and B.

Response in origional domain makes it difficult to distinguish between image classes A and B

Fig. 2 Optimal Discriminant properties of PCA.

A similar technique [4], employing Karhunen-Loeve Transformations, has been used to encode human faces with limited success.

Unfortunately the classical implementation of this technique is computationally very expensive where the input domain is large and this renders it impractical for all but trivial images. However the work of Sanger [5], Oja [6] and others has shown that an approximation of PCA can readily be implemented in Neural Network topologies which are considerably more efficient if the size of the input domain is large and the number of principal components required is small.

Implementing PCA in Neural Network Topologies

Training in these networks is self organised and motivated by Hebbian type learning on a simple 'sum-of-weights' linear node. Unlike many self-organised training schemes, however, it is not a winner-take-all type network where only one node learns aspects of the input pattern. The feature learnt by a node is deemphasised prior to training on subsequent nodes within the network to induce orthogonality in the components learnt by each node within the network. These networks have been proved to converge so that the weights of each node approximate progressively smaller principal components of the data set. Coding is achieved by presenting an image to the trained network where the output of each node gives the proportion of the principal component learnt by that node.

Neural Networks - A Logical Persfective

The PCA Neural Network model discussed earlier, and indeed the vast majority of Neural Network paradigms, are based around analogue nodes whose output is a function of the weighted sum of the input signals. Such systems are dogged with hardware implementation problems as relatively few nodes can be connected within a network. For this reason it is unlikely that a PCA type coding scheme using analogue nodes will ever become a practical reality.

However, considerable research effort by Aleksander and others [7] has been invested in the development of Logical Neural Network architectures which do not suffer these implementation drawbacks. Logical Neural Network nodes consist of networks of relatively sparsely connected Boolean functions which sample a binarised input space. These nodes can be realised using existing RAM/VLSI technology and practical, real-time, parallel systems are a viable prospect. A typical system is shown in fig. 3.

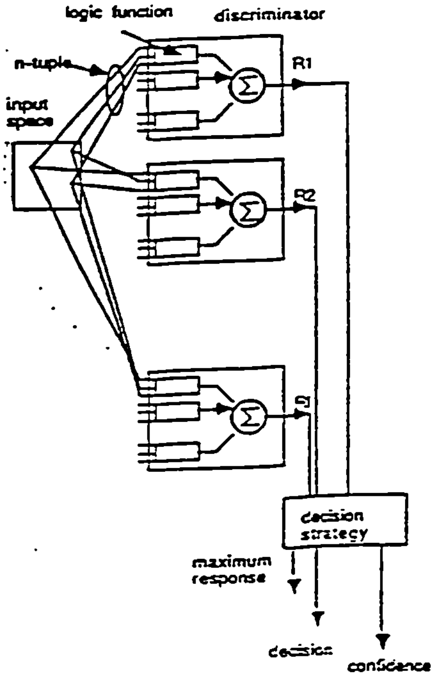


Fig. 3 A simple Logical Neural Network system

Each logical function within the net samples a group of binary elements within the input space - this group of samples is called a tuple. Every combination of binary elements within the tuple address individual locations within the system memory. The memory requirement of these systems grows exponentially with tuple size and consequently the nets are characteristically sparsely connected with tuple sizes typically varying from between 2 and 8. A system may contain many of these functions which sample the input space where the response of each function is combined in a discriminator to form a node which can sample a large input domain. If properly mapped onto the pattern space, higher tuple sizes will elicit increasingly higher order statistics from the training data which will result in a more efficient code.

Data is presented to the network during training and the sites within the logic functions addressed by the input tuples are stored within the net memory. As training proceeds a generalised representation of the class is built up within the sites of the system memory. It is this ability to generalise that gives these architectures powerful recognition properties. An 'unseen' pattern will evoke a high response from the system if it shares the same features which characterise the the training data for that discriminator.

However, adopting a logical node implementation of the PCA paradigm means that the mathematical framework which provides proof of convergence to the principal components is no longer applicable. Indeed, the mathematical formalisms which are employed to analyse the behaviour of analogue nodes are not readily transferred to logical node analysis and descriptions of Logic Neural Network dynamics are typically very illusive. Thus, the type of learning in a logical node can no longer be said to be directly equivalent to PCA but instead is motivated by the same factors which enable significant discriminant features to be extracted from the data set.

Tuples and System Performance

The tuple size can have a marked effect on the performance of the net and its ability to generalise. A low tuple size restrict the number of features that may be represented within a function a good generalisation is achieved through a coarse representation of the pattern features. Because of this reduced functionality the net has fewer sites to fill during training and the network is more prone to saturation where information previously trained into the net is overwritten as training proceeds.

Meaningful real-world images contain much correlated local information with respect to local feature entities within the image (such as low variance luminance levels and continuous line segment information , for example) which can greatly assist in reducing the dimensionality further, resulting in a more efficient code. This information can be extracted by mapping the tuples locally similar to the retinal fields found in biological visual processing topologies. The mapping of logical nets is often random because those correlated artifacts with low variance across the data set tend to promote saturation during training but this shortcoming can be easily circumvented.

An indication of the coding advantage afforded by local mapping of the tuples can be investigated by measuring the entropy of each function across the data set. Entropy gives a measure of the correlation between the features monitored by that function and is given by:

$$H = \sum_{i=0}^{\text{across func}} P_i \log_2\left(1/P_i\right)$$

Where P_i is the relative frequency of occurrence of a function within a tuple across the data set.

A low entropy for that tuple means that there is a poor spread of features across that tuple and indicates that it contributes little discriminant information. Fig. 4 shows that local mapping can, potentially, produce a more efficient code. It indicates too that the improvement afforded by higher tuple sizes is more marked when the mapping is localised. This highlights the fact that higher tuple sizes extract higher local correlations from the data.

Although the correlated data extracted through local mapping does indeed produce a more efficient code it can also lead to a poor distribution of features during training. This means that a narrow range of functions prevail throughout training and these dominate the response of the net whilst providing poor discriminant information. Entropy provides a measure of the usefulness of the features monitored by the tuples and enables us to bias then accordingly during training.

One way of implementing this is to train only on those features with a sufficiently high entropy. Fig. 5 shows those areas of high entropy for a data base of 500 binarised facial images. It is intuitively appealing that areas of potentially good discriminatory data for this data base form the outline of a face. It should be noted, however, that isolated tuples of high entropy do not *per se* guarantee that effective discriminant features will be learnt by that node.

The features are only optimal if there is a strong correlation across both the image and the data base. This scheme has the advantage in that it can substantially reduce the number of features to be learnt ,which lowers system memory requirements, whilst increasing the convergence time during training.

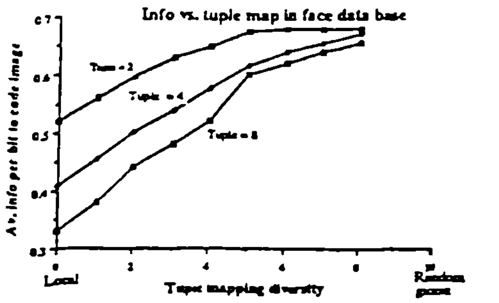


Fig.4 Variation of average information required to code an image and tuple size.

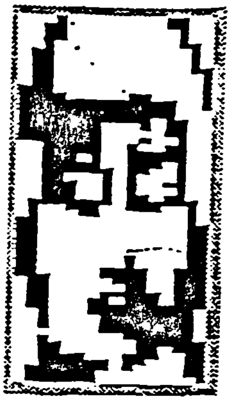


Fig.5 Areas of high information content for images across a data base of faces.

Preliminary results

At this stage of the work the training procedure did not automatically extract discriminant features from within the data base. Rather, each node was trained on a seperate exemplar image selected manually from the data base. Training amounted to presenting several shifted versions of the image to the assigned node, similar to the training scheme used in [7]. This formed a generalised internal representation of each exemplar within the trained node. After training, every image within the data base was presented to the nodes. The output of each node corresponds to the proportion of features that the image shares with the generalised exemplar image.

Each image within a data base of 500 32 x 59 binarised facial photographs was encoded by presenting it to an array of 8 trained networks. The output from the networks was ordered and a 3 bit ranking index attributed to each element of the code so that the images are represented as 24 bits codes.

An image from within the data base was corrupted both with noise and meaningful artifacts (a change in hair style, for example), coded, and matched against those in the data base. The matching strategy retrieved those images with the closest rank. The results are shown in Fig. 6.

Conclusions

Neural networks have the potential to provide very powerful coding mechanisms which permit rapid fuzzy matching of images for data base retrieval. An important adjunct to this work is the development of a localised self organising network to extract important image components which characterise the contents of the data base. Such schemes have been successfully implemented using analogue nodes and work is underway to apply similar learning rules to logical nodes.

As it stands the system described is intolerant to changes in scale, position and rotation. Neural network topologies have been shown to perform invariant recognition but it is unlikely that such networks will cope well with high dimensional images [8]. Gabor filters have been used very successfully as a pre-processor to impart a degree of independence to scale, position and rotation artifacts for invariant recognition tasks [9]. These filters exhibit some of the characteristics of the processing structures found in mammalian visual systems and hold much promise as a front end to the Neural Network coding scheme described here.

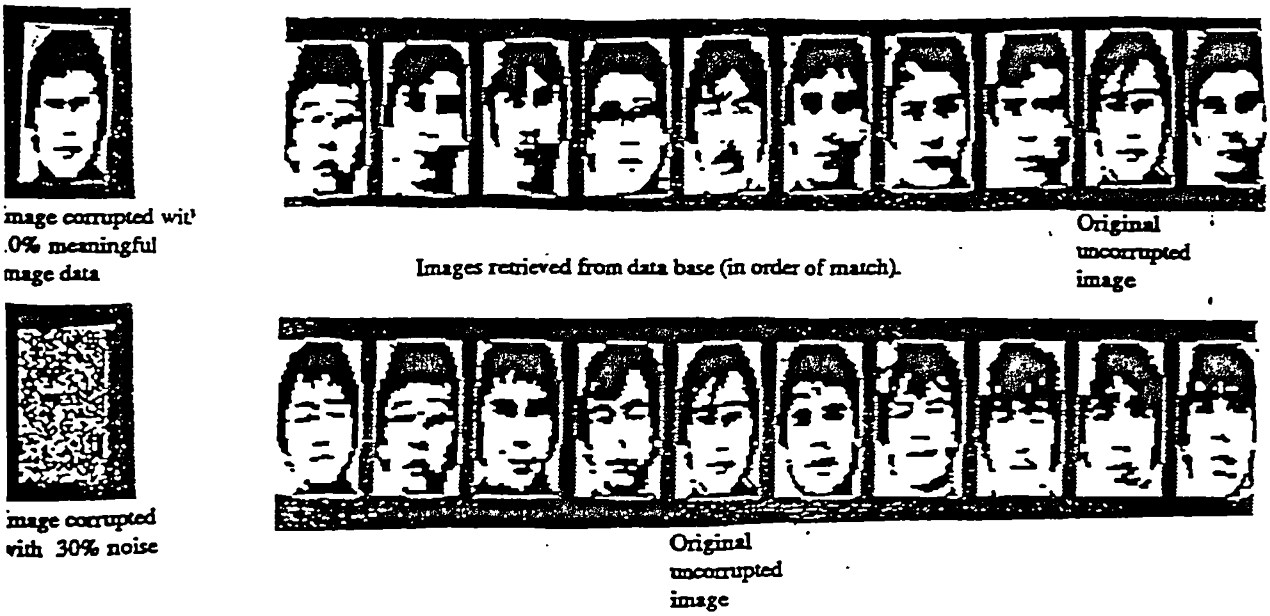


Fig. 6 Matching corrupted image with origional in a data base of 500 facial images

This research was supported by British Library Research Contract No. MO921.

References

[1] *Managing Image Data Bases.* IEEE Computing Magazine December 1989 pp. 3 - 62.

[2] Shi-Kuo Chang '*Principles of pictorial information systems design* '. Prentice Hall International.

[3] E. Oja '*Subspace methods for pattern recognition* '. Research Studies Press.

[4] M. Kirby, L. Sirovich '*Application of the Karhunen-Loeve Procedure for the Characterisation of Human Faces*'. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.12 No.1, Jan '90.

[5] T. D. Sanger ' *Optimal unsupervised learning in a single layer linear feed forward Neural Network* '. Neural Networks Vol. 2, pp 459 - 473.

[6] E. Oja '*Neural Networks, Principal Components and Subspaces*' .Int. Journ. Neural Systems Vol.1 No.1 pp.61 - 68.

[7] I. Aleksander, T. J. Stonham '*A guide to pattern recognition using Random Access Memories* '. IEE Proc. on Comp. and Digital Techniques 79 Vol.2, pp 29 - 36.

[8] K. Fukushima, S. Miyake, T. Ito ' *Neocognitron: A Neural Network Model for a mechanism of visual pattern recognition.*'

[9] E. Oja '*Distortion tolerant feature extraction with Gabor functions and topological coding* '. pp 301 - 304. Proc. conf. INNC Paris July '90.