Knowledge based dynamic pattern recognition: the recognition of dynamic patterns from minimal [i.e. minimal] examples

Justin Crowley
University of Wollongong

Recommended Citation
Crowley, Justin, Knowledge based dynamic pattern recognition: the recognition of dynamic patterns from minimal [i.e. minimal] examples, Master of Computer Science thesis, Department of Computer Science, University of Wollongong, 1996.
http://ro.uow.edu.au/theses/2308
NOTE

This online version of the thesis may have different page formatting and pagination from the paper copy held in the University of Wollongong Library.

UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.
Knowledge Based Dynamic Pattern Recognition:
The recognition of dynamic patterns from minimal examples.

Justin Crowley
November 1996

Department of Computer Science
University of Wollongong
I hereby declare that I am the sole author of this thesis. I also declare that the material presented within is my own work, except where duly acknowledged, and that I am not aware of any similar work either prior to this thesis or currently being pursued.

Just  A. Crowley
Abstract

This thesis introduces a dynamic recognition neural network model (DRNNM) that provides a theoretical basis for the resolution of a number of pattern recognition problems. These problems consist of: The Binding Problem; The Correspondence Problem; The Learning Complexity Problem and The Knowledge Transference and Extension Problem. The Thesis also addresses related issues, such as: recognition with scarce training resources; dynamic feature extraction and a methodology for reducing learning conflict or crosstalk.

The model consists of seven independent modules that communicate with each other to extract the maximum amount of information from the available data so as to provide more robust recognition. This objective has been achieved by designing each module for a different aspect of the pattern recognition process.

The testing and evaluation of this model was conducted predominantly from within an image recognition context. Where appropriate, the model has been contrasted with existing recognition systems, to highlight its unique characteristics and qualities. This has revealed the DRNNM's advantages in handling dynamic, variable data with minimal training examples, improving general recognition confidence of both digit and face recognition systems by 30%. This allows us to conclude that it is possible to recognise dynamic data with only a few static examples when using the DRNNM.
Acknowledgments

I would like to thank my supervisors, John Fulcher and Ming Zhang, for their interest, persistence and direction throughout the evolution of my thesis. In particular John for his perseverance and detailed appraisals of my numerous drafts and Ming for his technical reviews of my various ideas. I would also like to thank Ted Dunstone, for the numerous conceptual discussions, and the interest he displayed in my research. I am also grateful to my family, for their understanding and consideration through one of the more interesting periods of my life.

This thesis would have been considerably more difficult without the assistance and support of The Computer Science Department, The Center for Information Technology Research (CITR) and The Illawarra Technology Corporation (ITC). There help is sincerely appreciated.

I gratefully acknowledge the financial support of SITA (Societe Internationale de Telecommunications Aeronautique) for funding the research into face recognition, under the management of Beth Jackson.
Contents

Abstract ........................................................................................................................................ iii

Acknowledgments ................................................................................................................ iv

Chapter 1 - Introduction ............................................................................................................ 1
  1.1 Introduction ........................................................................................................................ 1
  1.2 Research Approach ......................................................................................................... 3

Chapter 2 - The Need for Static Based Dynamic Recognition Neural Networks ................. 4
  2.1 Background ...................................................................................................................... 4
  2.2 The Binding/Location Dependency Problem ............................................................. 5
  2.3 The Correspondance Problem ..................................................................................... 17
  2.4 Learning/Training Problems ....................................................................................... 20
  2.5 Practical Limitations ..................................................................................................... 27
  2.6 Conclusion ..................................................................................................................... 28

Chapter 3 - The Theoretical Basis Of The Dynamic Recognition Neural Network Model (DRNNM) ......................................................... 29
  3.1 Introduction ................................................................................................................... 29
  3.2 Theoretical Overview ................................................................................................... 31
  3.3 The Dynamic Recognition Neural Network Model ................................................... 41
Chapter 1 - Introduction

1.1 Motivation

The main motivation for this research is the inadequacy of existing pattern recognition paradigms when required to recognise dynamic data from static representations. In general, the models available are only conducive to the recognition/classification of simple problems. Difficult dynamic recognition problems, where recognition is dependent upon minimal training examples and non-linear associations, such as human face and handwritten digit/character recognition, require the development of more complex models capable of accommodating considerable data variability. It is not possible to effectively recognise this class of problem by simply extending or combining together simple classifiers. This is because present recognition models assume that:

a) the training data will provide an adequate representation of the pattern distribution and variability [Valiant] [McMichael] [Radford]

b) it is possible to find invariant mappings allowing associations to be made between various pattern instances [Cottrell 90] [Chen Ching-Wen] [Craw 92b] [Perry] [Kosugi] [Li Chihwen]
c) the system is capable of substantial generalisation enabling consistent recognition in adverse conditions [Akamatsu][Allinson][Bouattour][Kerin][Kirby 87][Turk]

d) it is possible to find features that allow novel pattern variations to be accommodated by aligning these features with learnt pattern examples [Abe][Brunelli] [Gay] [Gordon] [Jia] [Sakai] [Tsui][Malsburg 92][Ullman][Bashri]

However, other than in the most trivial problems, it is unlikely that any of these conditions will hold. There may only be a small number of examples of a pattern class, or the training data may only represent a small fraction of the possible relevant data. Alternatively, unanticipated class variations may complicate feature extraction and classification. These problems prevent the formation of suitable data associations, making robust recognition difficult.

Thus, the aim of this research is the development of a model/theoretical basis for the robust recognition of dynamic patterns from single/minimal static representations. To achieve this objective it is necessary to determine what the fundamental inadequacies of existing models are, and whether it is theoretically and practically possible to develop a model that can resolve these problems. It then becomes necessary to determine if it is possible to recognise dynamic data from single static representations and to determine the necessary foundations for this to be accomplished. Ideally, this should enable the construction of a practical implementation that demonstrates the research objectives.
1.2 Research Approach

To achieve this aim:

Chapter 2 - discusses the inadequacies of existing models and theories - elaborating on the issues that need to be addressed for more robust recognition and providing a basis for future model development.

Chapter 3 - provides the theoretical foundations of this research - determining the theoretical properties required.

Chapter 4 - details general implementation aspects - this involves the practical realisation of the theoretical properties discussed in chapter 3, providing a means for comparison with existing systems/models.

Chapter 5 - provides specific model implementations and results from experimentation with different dynamic pattern recognition problems.

Chapter 6 - discusses and evaluates this research, highlighting the more significant findings.
Chapter 2 - The Need for Static Based Dynamic Recognition

Neural Networks

2.1 Background

Recent developments in Developmental Psychology suggest that the recognition of patterns is linked to the context in which the pattern is experienced [Rota] [Kellman]. For example, Perceptual Category Theory argues that the recognition of handwritten characters is dependent upon having the experience of writing, or mentally tracing, the desired character’s outline [Reeke]. Recognition consequently becomes a dynamic process where relevant eye-motion is related to particular features [Rybak 91] [Birnbaum]. This process is particularly evident when we are presented with difficult characters to recognise. These characters lack the visual simplicity and consistency of typewritten characters, and therefore cannot be easily recognised using template based operations - that is, direct comparison. This necessitates that we try and recognise these characters by determining how the character was written - finding the character’s end/starting, or intersection points and visually, and mentally, tracing the character’s perceived formation. This information is then normalised and related to our own character understanding.

Unfortunately, with static recognition information regarding the formation of the recognition pattern is not available. The pattern is simply there; the recognition system having only access to a snapshot - a brief pattern instant that has no past,
future or external existence. This increases the difficulty of the recognition problem as it is no longer apparent what is being recognised; the pattern and its supporting structure being interrelated.

In effect, the loss of the data’s dynamics causes the pattern data to be tightly bound to the data instance’s context. This hinders learning, and complicates recognition, making it difficult to determine what needs to be learnt, or recognised. This predicament can be related to Quine’s riddle regarding the difficulties with language translation [Quine]. In this story you have just arrived in a strange land and are required to learn the natives’ language. The problem is that if you do not perceive the world in the same manner as the natives, then when a rabbit hops by and a native says “rabbit”, you are unable to determine whether the native is saying “hopping”, “white”, “furry”, “long ears”, “rabbit”, “rabbit parts”, all of these, some of these, or none of these. You are unable to associate the native’s exclamation with the appropriate context. In automated pattern recognition, the recognition system is generally in a similar situation - it does not perceive, or have any knowledge of the recognition problem domain and consequently cannot associate the data with a particular context to derive what needs/should be learnt.

2.2 The Binding/Location Dependency Problem

2.2.1 Definition

The inability of a system to differentiate between different contexts is referred to as the Binding Problem. This refers to the binding of the context and the recognition data together, making it difficult to determine what needs to be learnt. The effect of this is that the classifier develops a data location dependency, that is, during learning some
data locations develop greater significance. This occurs because the values at these locations, and their associated weighting, are what the system uses to partition the data into its respective categories. However, this location dependency means that subtle changes in data values can have significant effects on data classification.

Ironically, binding is essential to reduce the possible solution search space. Without binding classification would be computationally infeasible. If the pattern was not bound to the classification surface then the recognition system would in effect be trying to solve a graph matching problem [Tsotsos] [Pentland 92] - each data item would have to be associated with each other data item. A comparison of two data samples would consequently require interaction between possibly every data item in both samples.

![Diagram of Binding Problem Illustration](image)

**Figure 2.1 Binding Problem Illustration**

Figure 2.1 illustrates how the associations a pattern classifier develops during training, with the “Training Data”, are not necessarily applicable to a novel data instance. During
the training process the classifier creates an association between the training data and specific locations upon the medium through which the data is expressed. If the "Recognition Data" is in a different relative location, then activation of the learnt classifier regions does not occur, resulting in a reduction in the classifier's response and recognition confidence.

It is unlikely that the rigidity the Binding Problem introduces in a recognition system is desirable. In some cases where fine discrimination is needed, such as the presence of cancerous growths in the context of medical imaging, the Binding Problem provides a potential means of easily distinguishing different conditions. However, where there is considerable intrinsic variability within the recognition class, the rigidity limits recognition confidence, placing greater emphasis on the classifier's generalisation ability.

We can consequently regard classifier generalisation as being the amount of change required before the effects of the Binding Problem become evident. On this basis, the Binding Problem is dependent upon:

1. How well the system has been trained. The problem of over training / over fitting that occurs in some systems, for example Multilayer Perceptrons (MLP's), arises because the system has "homed in" on a few locations and become too specific in its classification. A very small change may consequently affect pattern classification. This may involve as little as one or two locations. For example, there was a rumored neural network system that mistakenly discriminated between male and female subjects, on the basis of the amount of white space separating the tops of the
subjects' heads from the upper boundary of the picture. This type of problem is particularly nasty in that it can be difficult to diagnose.

2. The type of data being classified. For example, if the data being learnt is a submanifold, then problems can arise from something as simple as a pattern translation. The translation changes the pattern's relative location within the larger pattern context and consequently if the system training does not allow for this type of variation, or if appropriate constraints have not been enforced, then the system will not recognise the class instance.

3. The way in which the classifier has been trained. The sequence in which a classifier is presented data (MLP's), or the frequency with which different class examples are presented (Learning Vector Quantisation - LVQ and Self Organising Feature Maps - SOM or SOFM) can potentially adversely affect the weighting of different locations. Consequently, the observable effects of the Binding Problem may be obscured, or enhanced, by different classifier training methods, or models.

The negative aspects of the Binding effect can be observed in the following graphs.
Effect of Translation on Digit Recognition Confidence

![Graph showing the effect of pixel translation on digit recognition confidence.]

Figure 2.2 Effect of Translation on Digit Recognition Confidence.

Effect of Rotation on Digit Recognition Confidence

![Graph showing the effect of two-dimensional rotations on digit recognition confidence.]

Figure 2.3 Effect of Two Dimensional Rotations on Digit Recognition Confidence.
Figure 2.4 Effect of Scale Increases on Digit Recognition Confidence.

Figure 2.5 Effect of Translation on Human Face Recognition Confidence.
Figure 2.6 Effect of Two Dimensional Rotations on Human Face Recognition Confidence.

Figure 2.7 Effect of Scale Increases on Human Face Recognition Confidence.
Each of the above figures depicts the effects of different domain related holistic transformations on Digit (figures 2.2, 2.3 and 2.4) and Human Face (figures 2.5, 2.6 and 2.7) recognition. The character size used was 32x32 pixels, with the Face data being 64x64 pixels. The figures indicate that with the tested systems even relatively slight transformations have a significant effect on recognition confidence. The digits were tested using standard Nearest Neighbour classification on 10 sample digits. The faces were tested using the same classification method as the digits and contrasted with the EigenFace approach [Pentland 91]. 70 individual’s faces were used from a locally created database.

The digit recognition results corresponded with equivalent tests on character recognition by Alexandre and Guyot [Alexandre 95]. They used a weight sharing MLP, similar to LeCun’s model [LeCun 90], consisting of an input layer (the image), three hidden layers and an output layer of 26 units for the 26 characters. They had good recognition results for centered characters, 99%, but as soon as the characters were translated more than 2 pixels, character recognition plummeted to 55% and progressively lower as the translation size increased.

### 2.2.2 Traditional Solutions

The traditional solution for the binding problem is to:

1. use local/structural features that can be extracted from the pattern. These features and their relative locations can then be compared individually with a feature database and then globally with the training database. The advantage of this
approach is that the local features provide [Malsburg 87] [Rybak 90] [Craw 92a] [Yuille] [Nakamura 91a] [Sakai]

a. A tolerance to data shifts - minimising local topographic changes.
b. Data redundancy, by overlapping local feature content.

1. use a large number of examples that characterise the problem. This would consist of training the system with as many diverse examples of a class as possible, to try and evenly distribute data samples throughout the entire solution space of the problem. [Fahlman 90][Yan][Oja]. However, this introduces problems of its own, such as:
   a. selecting the optimal number of class examples.
   b. ensuring that sufficient resources are allocated.
   c. being able to find sufficient examples
   d. updating classifier knowledge
   e. minimising classifier conflict
   f. avoiding loss of previous learning

These problems are all interrelated and require an intimate knowledge of the problem domain and the classifier characteristics. They are discussed in more detail in the following sections.

2.2.3 Local Feature problems

A disadvantage of local feature based recognition approaches is that they can take considerable time to perform the required local and global comparisons. If there exists a large number of possible local features, all of these features need to be evaluated before a local feature can be classified. The time required for this initial stage can potentially
be reduced by using tree based classification techniques [Sanger] [Sankar] [Hunt] [Quinlan] [Breiman] [Zhang & Fulcher] or hierarchical based indexing schemes [Crowley 94b]. However, the whole pattern still has to be evaluated; Graph-Matching techniques have proven effective in this respect, but are notoriously slow [Malsburg 92].

A method developed by Rybak [Rybak 91 & 92] based on associations between eye-fixations and feature retrieval, has demonstrated a possible way of decreasing the computational load. This method selectively reduces the feature set by taking into consideration possible gaze behaviour. This approach could potentially be adapted to other classification problems, assuming suitable "fixation" points could be determined, for example, end and intersection points in character recognition, minutiae in fingerprint recognition [Leung], or characteristic facial features in face recognition [Kobayashi]. However deciding upon what attributes the features should have and finding these feature points can be difficult. This predicament is discussed in section 2.3 - The Correspondence Problem.

Local feature based schemes also have to address how to handle different instances of a local feature. For example, will a rotated line be "a line that has been rotated", or "a rotated line"? If it is to be "a rotated line", then some example of a rotated line will need to be stored in the feature database. Consequently, the feature database may grow extremely large, especially within a complex pattern environment. This means that the system will need more time to search for the appropriate local classification. Alternatively, if the line is to be classified as "a line that has been rotated", then some mechanism will be needed to allow the recognition of the line as being rotated. This might involve a procedure that automatically rotates and classifies the feature.
Interestingly, in some respects local systems are just as vulnerable to the binding problem as their global equivalent. A local feature is just as bound to its context as a holistic feature. However, a local feature can be provided with extra mobility, within the pattern context, allowing more flexibility, so that adjustments can be made for some data variations.

A major consideration with local features is that their relatively small data representation tends to reduce the gestalt effect that is such an intrinsic part of holistic/global recognition. All, or at least reasonable percentages, of the local features have to be reconnected/reassociated for a final classification. This process is very dependent upon the local feature classification component of the recognition process. As the local features are only representative of very small parts of the overall pattern, local systems do not have sufficient information to take into consideration subtle holistic pattern changes. For example, consider a person wearing glasses. How can a local system, with only bits and pieces, effectively associate the various components as belonging to a pair of "Y" glasses? The local features will vary with different people, and consequently, consideration will have to be given to whether the glasses are occluding the person's face; or whether the person's facial details are hindering the perception of the glasses. Another example is a human face rotated slightly around the y-axis. All the original local features, obtained from a front face training database, will no longer be applicable. They will all be slightly out of alignment. The problem is, whether the change in these features is sufficient for features to be mistaken for other features, and thereby result in a misclassification. This lack of holistic awareness in local systems makes it difficult for these systems to take into consideration holistic
changes such as glasses or rotations, and therefore hinders their global classification effectiveness.

### 2.2.4 Alternative Solution

In Object Vision the binding problem is often compensated by using the Alignment Method. In this approach recognition is viewed as simply the matching of a data pattern with a series of stored patterns. If we let $P$ represent a set of stored patterns and $D$ the pattern to be recognised, we can argue that in general, $D$ will differ from all “known” patterns instances, due to some transformation $T_i$ that has reduced its similarity with the “known” patterns in $P$. Therefore, to make allowances for these transformations, we can adjust $D$ to try and increase it's similarity to a pattern in $P$. If $T$ is the set of available transformations, then the adjustment of $D$ will require the selection of an appropriate transformation $T_i \in T$, such that the known patterns $P_i \in P$, and the transformed $D$ are as similar as possible. This procedure reduces the complexity of object recognition by systematically aligning the viewed object with the known object representations. Recognition is consequently reduced to establishing a correspondence between the recognition data and the known data characteristics [Kellman 91] [Ullman][Basri].

Unfortunately this approach overlooks a number of significant problems. How does one:

1. represent the different patterns instances, $P$.
2. determine the appropriate transformations, $T$.
3. decide which transformation to use to minimise the difference between the recognition pattern $D$ and the known pattern $PT$.

a. align lower dimensional spaces within a higher dimensional space, e.g. image recognition problems

1. accommodate data occlusion, where necessary alignment information is obscured.

2. align multiple pattern classes?

These problems have tended to limit alignment methods to a small subset of Object Recognition problems - usually isolated, single model, wire-frame objects.

2.3 The Correspondance Problem

It could be argued that the main problem with recognition is not so much the binding of the pattern's discriminatory information to particular locations, but the inability of these systems to relate the pattern information to the training examples or representative class models. This is commonly referred to as the Correspondance Problem.
Figure 2.8 An example of the Correspondance Problem

As data becomes more complex the determination of suitable recognition features becomes more difficult. This is because the features become more troublesome to define and harder to consistently locate. This hinders the development of feature associations making the data more difficult to classify. For example, in the handwritten character recognition problem of figure 2.8, which features do we use to associate and discriminate different character representations? Do we try and define all possible line shapes? Do we base our classifications on the presence or absence of line intersections, or do we extract some other type of features? After classifying all these features, how then do we associate them? Do we simply compare the presence or absence of features, or the relative distance and direction that separates features?
Even if all these problems are suitably solved they are generally application specific with little benefit to other recognition problems.

The Correspondance Problem is most evident in Local Recognition Systems, as they are very dependent upon the consistent location of characteristic features. For example, in Face Recognition, Rybak used edge intersections to limit feature searches, Malsburg utilised feature endpoint differentiation to find interesting points, whereas Crowley [Crowley 94b] used a structured, mean variation approach to determine initial search areas. Other approaches extract location feature information on the basis of texture variations [Nakamura 91a], or structural features [Yuille] [Allinson] [Ellis 87] [Tsui]. If this information cannot be consistently retrieved then the recognition system will perform poorly. In these approaches classification is dependent upon positive correlations between the selected and feature database regions to reduce recognition time and to ascertain class membership. For example, the Alignment Method is very dependent upon distinctive features for the development of associations between recognition data and model prototypes. In Ullman's [Ullman] case, correspondences between three or more points on the recognition object and the appropriate model prototype are needed for object alignment. In noisy, occluded, complex, real-world environments, establishing these relationships can be difficult, as the appropriate points may be hard to find. Therefore, although the Alignment Method does provide a potential solution to the Binding Problem the Correspondance Problem makes it difficult to implement effectively.

Knowledge based approaches such as flexible wire frame [Reinders] models and the MBASIC model [Aizawa 89] are also affected by the Correspondance Problem. These
approaches associate pre-defined models of the data domain with the pattern features. This allows pattern variations to be rapidly accommodated. At present these methods are mainly used in well-defined motion estimation applications, such as video conferencing, to reduce computational demands. However, if the model feature points cannot be located then these systems will not operate effectively. Consequently, manual point selection is often used [Aizawa 93]. For example, if we try and normalise a person’s face by using the location of their eyes and mouth [Aizawa 89] [Yuille] [Craw 92b] [Basri] [Sakai], how do we cater for inaccuracies? What is the effect of 3D rotations on the relative associations between the eyes and mouth locations? More importantly how do we consistently locate eye and mouth regions in complex variable environments? We can use deformable templates, or snakes [Welsh 91] [Yuille], but what happens then if the person blinks or is wearing glasses? Will our system still find the features reliably, or will a lot of time be wasted in trying to find features that no longer exist in a recognisable form?

2.4 Learning/Training Problems

It could be argued that all recognition problems are related to the training and development of the system.

2.4.1 Generalisation

Generalisation is a term commonly used to refer to a system's ability to classify novel pattern instances; the better a system's generalisation ability, the greater the amount of change or pattern variation the system can tolerate. In general, if suitable training examples have been used and appropriate characteristic features learnt, then even very simple systems will generalise reasonably well. What distinguishes a good system is its
ability to generalise even under adverse conditions, where data and resources are scarce. For this reason, classifier research is largely concerned with finding more effective means of extracting better features so as to improve generalisation.

Unfortunately generalisation, in a lot of cases, is nothing more than educated guessing. A recognition vector is simply compared with the training vectors and a classification made on the difference between the training and recognition vectors. If this difference is less than an arbitrary threshold, then the recognition vector belongs to the same class as the comparative training vector. If the recognition vector belongs to more than one class, a vote may be taken to determine which class had the greatest number of successes, or the class with the closest member may be selected. How well the system generalises consequently depends on the size of the similarity threshold, and has absolutely nothing to do with the application of domain specific knowledge or understanding.

This makes it difficult to compare different models as variations in recognition accuracy are generally related more to the type of problem than the actual features selected.

For example, consider figure 2.9:
Using a direct comparison, “Comparison 2” is more similar to the “Initial Object” than “Comparison 1”, as “Comparison 2” and the “Initial Object” have an area in common. In both cases the recognition scores for “Comparison 1” or “Comparison 2” would be well below any similarity threshold measurement. This is just an illustration, and would not be a recommended procedure to use for the recognition of these classes, although some researchers have used similar approaches in character recognition studies.

What Figure 2.9 demonstrates is how the Binding Problem affects generalisation. No matter how well the system is designed, if it does not provide some means of relating the different instances, it is possible for situations to arise where very similar instances are ignored in preference to less appropriate instances, as consideration has not been given to the recognition domain’s dynamics.
In dynamic recognition the normal classifier forms of generalisation are not sufficient. Different data instances may be very similar but still a large Euclidean or other location based measuring system, distance apart.

The only reliable means of resolving this problem is to have knowledge of the recognition domain. This cannot be obtained by simply providing numerous examples. If anything this just creates more problems, such as learning interference.

2.4.2 Knowledge/Learning Interference: Crosstalk

As a system learns, previous knowledge can become lost or confused with more recent learning. This necessitates the continual relearning of old information to ensure nothing important has been lost. This amnesia is most pronounced in systems that have not allocated enough resources for the amount of information they are trying to learn (not enough hidden units (MLP), or code-book vectors (Nearest Neighbour, LVQ)). This is exactly the sort of situation that might occur if trying to train a system by providing vast quantities of examples with little consideration for their apparent differences.

To minimise learning interference there has been research into developing systems that grow dynamically, and in methodologies for determining the ideal number of units needed for effective learning. The system developed by Fahlman: Cascade Correlation [Fahlman 90] and other tree-based approaches [Sanger] [Zhang & Fulcher] grow in response to the system's classification needs. Determining the optimum number of hidden units is a popular area in MLP research and is indicative of the difficulty in finding the appropriate network size.
Learning interference is of particular concern when extending an already trained system [Jacobs]. Extending a system to cater for new knowledge can be extremely difficult and time-consuming without introducing learning interference problems. The system has to be re-exposed to previous data as well as the new information. In some cases, such as with MLPs, the whole system may have to be totally retrained. This may even occur with dynamic tree-based systems if the extensions introduce significantly new properties. This highlights a structural limitation of these approaches.

An alternative solution to system amnesia/learning interference is to simplify the learning task. The design of Modular Neural Systems [Poggio] [Smotroff] [Murre] [Jacobs] is one means of achieving this objective. These systems divide the learning task into independent modules and thereby reduce the overall learning complexity. The problem is determining how to divide the learning task into discreet units, and determining the functionality of each module.

### 2.4.3 Knowledge Transference

Ideally, a system should be able to transfer knowledge from one pattern recognition problem to another. In this way the time and effort expended on the development of a recognition system might be useful for more than one particular problem. Unfortunately, "knowledge" is generally very specific and cannot be used for other recognition tasks. This is mainly because the knowledge gained during training is embedded in a complex, intricate lattice of weights (MLP, MAC); task specific
codebook vectors (LVQ, SOM), or recognition invariant features (Neocognitron) [Fukushima 89]. Although considerable research has been and is being conducted in finding ways of maximising/reutilising the knowledge stored within a system's weights, little benefits have arisen so far [Carbonell] [Fahlman 87] [Fozzard] [Sestito].

A more logical approach would be to store the knowledge independently of the recognition process. This would allow this knowledge to be accessed by other systems when appropriate. This is an important aspect of Inductive Learning - the learning of generalisations from an example set [Honavar]. This enables theoretical separation of the environment from the recognition data, but also creates other problems. How do we:

1. implement this approach within a real-world situation?
2. determine when and how to use this knowledge?
3. restrict the learning possibilities to reasonable limits?

In general, a priori knowledge and/or assumptions have to be made about the data for effective learning. However, this Inductive Bias restricts the system reintroducing the earlier problems with training data [Geman] [Bienenstock].

2.4.4 Speed/Difficulty

The rate at which a system learns is generally speaking proportional to the difficulty of the learning task. We can consider the difficulty of a recognition task as being related to the complexity of the decision regions formed by a system during training. The more convoluted or intertwined the classification manifolds, the more likely that the learning
process is conceptually and computationally complex. Although if similar classes are being classified differently, then the search for suitable discriminatory features may be time consuming.

This introduces the problem of determining what has been, and is being learnt by the system. As the complexity of a classifier's learning increases it becomes more difficult to ascertain what has been learnt other than through empirical tests. This is important in that the more difficult the search, the greater the chance that the system will learn obscure or trivial features.

Another area of concern is that the global minima/optimal solution for a system is not necessarily the best global minima/optimal solution for the recognition task, just for that particular model/system under the training conditions. The global minimum may have been found, but there is no guarantee that the features used will also be effective with novel data.

Other areas of concern include:

1. How does the system learn if only minimal data is available? This is especially important if the system is being used in a dynamic environment and therefore is dependent upon having a wide variety of examples to characterise the problem.

2. What are the consequences of not providing a sufficient variety of examples? Will the system's performance gracefully degrade, or will it simply misclassify novel instances? In some cases, the possible negative connotations can be reduced by varying the tolerance of the system e.g. any Nearest Neighbour
Classifier derivative. However, this may make the system's recognition so specific that its effectiveness is reduced.

2.5 Practical Limitations

The availability of a sound theoretical framework does not always guarantee that models based upon this structure will function effectively in novel circumstances. For instance, it is theoretically possible for a three-layer backpropagation neural network, with an infinite number of nodes, to solve any arbitrary mapping problem within a compact set [Lippman] [Kearns]. This does not mean that all recognition problems have now been solved and that pattern recognition is now a redundant area of research. There still exist problems with determining the optimum network size, even though there are several dynamic Neural Networks approaches that can adaptively adjust these quantities [Fahlman 89]. It is an unfortunate fact that classification systems do not understand recognition problems, they simply find data associations. Therefore, the training examples need to cover the entire parameter space for a classifier to learn and develop the necessary associations to classify different data transformations. Consequently, a classifier’s learning is always potentially inadequate, independent of its theoretical structure.

We can even go further and argue that real recognition problems do not generally form compact sets and are therefore not solvable using traditional classifiers in their present operational form. If we consider a compact set as a closed, totally bounded metric space, then we can ask does a recognition problem, such as Handwritten Character Recognition, form a compact set? If it does not then this would suggest that
traditional classifiers cannot totally solve this problem. We know that handwritten characters are generally restricted by image constraints and are therefore limited to a defined spatial area. However, a set is closed only if it contains all possible data. As handwritten characters are continually changing and evolving, varying from one person to the next, we can argue that it is not a closed set, as not all of the set’s potential elements are defined. At each instance in time there may exist more and different character variations that continually extend the initial character set. Therefore, although an infinitely sized Neural Network may be able to learn the first character set, is this learning still applicable to new, previously, undefined characters? This will depend on how well the system generalises (generalisation error), and how characteristic the initial training data was (empirical error). Ironically, such a large Neural Network’s success, or failure, may be difficult to determine as even small systems rapidly become difficult to interpret. There are simply too many interactions to easily comprehend their functionality.

2.6 Conclusion

There are a wide variety of problems with Pattern Recognition. These problems are largely due to a lack of suitable models/frameworks to organise and construct robust recognition systems. Simple classifiers are inadequate as they lack the necessary knowledge and capabilities to address the variability that exists in real problems. This limitation cannot be overcome by the arbitrary grouping of these classifiers into hierarchical, modular structures as they lack the necessary problem understanding to cater for novel pattern instances, or situations. Therefore, more complex systems need to be developed that take into consideration the data dynamics as a separate entity rather than as an aspect of the recognition system.
Chapter 3 - The Theoretical Basis of the DRNNM

3.1 Introduction

In this chapter the theoretical foundations of the Dynamic Recognition Neural Network Model (DRNNM) are presented. We discuss the properties of the functional units comprising the DRNNM and why this approach was developed to address the recognition problems discussed in Chapter 2.

The DRNNM was developed on the assumption that learning and recognition are dynamic processes. This belief is supported by considerable psychological evidence that suggests learning is a consequence of a desire for novel experience. Recent child development research [Thelen] indicates that children satisfy a desire for a certain level of neural activity by keeping interesting objects in sight. They therefore interact with their environment increasing their awareness of the implications of their actions. In effect, they establish a link with the external world through their existence within the world [Freyd83 & 92] [Edelman]. This causes associations to develop between the different aspects/instances of a pattern class and the time-dependent neural activity generated by the child's appreciation of that pattern class. We could say that the child learns and develops recognition skills, dynamically, through experience.

However, traditional recognition systems do not treat static representations of dynamic data in a dynamic fashion. They process the data statically. Therefore,
valuable data relationships are lost that may be difficult if not impossible for the system to learn. This forces a learning system to create class associations based only on data correlations. This is the equivalent of a child trying to learn while incapable of interacting with the world and experiencing the effects of its actions. The child will have no need, or incentive, to learn, or the capability of associating different events. Ironically, we would not expect a child to learn under these conditions, yet this is the type of environment that an automated recognition system is generally expected to learn within.

It is the detachment, or abstraction, of recognition systems from their environment that forces these systems to be dependent upon the validity of the training examples and the suitability of the learning framework. If there are training errors, or insufficient examples, then the system will become confused and learn ineffectively. The system has no means to verify, or validate, its learning experience, other than by the data it is presented with. Ideally, we could overcome this limitation by providing the system with access to a conceptual model of the problem domain. However, this is often hindered by the difficulty of developing models that encapsulate the probability and possibility space of high dimensional or complex data. This is particularly the case where non-linear class associations may exist. One of the original reasons for the high level of interest in Neural Networks was that they were perceived as modeless learning environments that developed the equivalent of a domain model as an intrinsic aspect of learning. Unfortunately, the networks were found to require prohibitively large training sets. Therefore, strategies evolved using a priori knowledge of the recognition problem to reduce the training sets to more manageable sizes. This is generally referred to as the bias/variance dilemma [Bienenstock], where in order to
accommodate data variance a priori knowledge of the pattern domain is used, biasing the learning process.

This thesis argues that for recognition to be effective the system should be able to interact with the data it is learning from. It should be able to experiment, and trial, different alternative solutions to a particular problem. It should have some means of appreciating the domain that the data it is expected to learn originates from. It may then be able to extract meaningful associations and perform robust recognition from minimal examples.

3.2 Theoretical Overview

How then do we enable our recognition system to interact with the data domain? The most obvious solution is to provide it with a means of appreciating the data domain that it is working with. This involves enabling the system to perceive and interact with the data it is recognising. In other words, the system needs to understand how different responses affect the data's presentation and have available courses of action that it can use to minimise and accommodate these effects.

This differs from the traditional perspective that recognition/classification is a form of mapping between the perceived data and the recognition class. Traditionally, for example, learning to recognise a character or face would involve the learner receiving a training set \( y = \{(x_1, c(x_1)), \ldots, (x_n, c(x_n))\} \) consisting of sample images \( x_i \in X \) where their correct classification is defined by \( c(x_i) \). In general, the examples \( x_i \) would be chosen from \( X \) with respect to some fixed probability distribution \( P \) on \( X \). The subsequent training set would be the only information that the learning system has.
available about the data. Therefore, for the system to learn the classification, \( c \), the system requires an \textit{hypothesis space}, \( H \), to provide a series of functions from which it can select the best approximation to \( c \). If the system learns effectively then a series of functions will be found that provide an invariant mapping that always correctly classifies the recognition data irregardless of its state.

Is the traditional approach realistic? Is it possible to find an invariant mapping between a data set and a series of classes? Is this, for example, how we recognise other peoples' faces from minimal examples? Do we have some mapping mechanism that provides us with the ability to automatically associate a person's face with the appropriate classification? Cognitive research suggests that this is not the case. Researchers have found that it does take a significantly greater period of time to recognise a face in an unorthodox position, such as upside down, than it does to recognise that same face in a normal position [Yin] [Shephard]. If an invariant mapping was been used then there would be no appreciable time difference with the recognition of either face. The time difference occurs probably because the facial instance is being synchronized with the normal facial representation, and consequently for faces that are in significantly different state than the norm more mental gymnastics are required to normalize the face and therefore more time is taken. This would also explain the tendency for people to automatically, physically rotate their head when trying to recognise somebody who is upside down - the physical adjustment of their visual system reduces the mental effort required [Yin]. Therefore, what is being observed with face recognition, and speech and character recognition, is an application of an individual's knowledge to the particular problem. Data variance is handled because the person has knowledge of the general types of
data variations and is capable of recognising these variations and making allowances for them.

Invariance and uniqueness are essential for reliable recognition. Traditional recognition systems are limited in their effectiveness in that even if they provide an invariant mapping that allows any variation of a pattern to be associated with a particular class; this is only achieved by sacrificing the pattern context. These systems function by finding a mapping, or pattern representation, that ignores the actual state of a pattern. This allows the recognition system to associate different pattern instances with a specific class. However, in so doing the recognition system cannot maintain knowledge of the pattern’s state without affecting it ability to associate the pattern with a specific class. In effect a static, or passive, recognition system does not recognize a pattern; it associates a pattern with a known class by destroying the pattern’s uniqueness. The unique quality of the pattern’s transformational state is lost. In some domains this information may be just as valuable as the class association. In a dynamic recognition system the pattern state and its class associations are maintained. A dynamic system searches for possible transformations that allow a pattern to be associated with a known pattern/class, thereby maintaining the initial pattern’s uniqueness and providing invariance through the transformation process.

It is important to understand that what makes recognition difficult is not the structure, or the dynamics of the pattern itself, but the difficulty in defining all possible pattern states. If a pattern only as one possible state, or if all possible states can be defined, then there is no recognition problem. As Garner [Garner 1974] suggests the challenge is specifying the set of other stimuli or identifying the alternatives that did not appear in any given trial. In other words, how well the system generalizes. Addressing this problem is crucial when minimal examples are used as almost every pattern instance will be different from the examples used to develop/train the system.

This thesis argues that a recognition system using only minimal examples is required to treat a pattern and its different representations as a dynamic system. Therefore each pattern instance represents a different pattern state, with each pattern state being
associated with other pattern states through a set of transformations. These transformations define the variability of the system and the type of pattern variations that can be addressed.

We can denote a transformation \( T \) by

\[ u \xrightarrow{T} v \quad \text{or} \quad v = T(u) \]

where \( u \) and \( v \) represent states of the system, with state \( v \) being the image of state \( u \) under transformation \( T \).

These transformations can be composed by the consecutive application of transformations, such as one transform being applied after another, for example:

\[ u \xrightarrow{T_1} v \xrightarrow{T_2} w \quad \text{or} \quad T_2 T_1 (u) = T_2 (T_1 (u)) = T_2 (v) = w \]

Generally, transformations under composition are not commutable, however, these transformations are bijective (one-to-one and onto), such that every state acts as an image of the mapping and is an image of a unique state. A characteristic of bijective transformations is that they are invertible and have an inverse mapping obtained by reversing the transformation, for example

\[ v \xrightarrow{T^{-1}} u \quad \text{or} \quad u = T^{-1}(v) \]

As an inverse mapping is also bijective, a bijective transformation and its inverse are called mutual inverses – they imply one another. The composition of mutual inverses gives rise to the identity transformation \( I \), such that:

\[ T^{-1} T = T T^{-1} = I \quad \text{or} \]

\[ (T^{-1} T)(u) = T^{-1}(T(u)) = T^{-1}(v) = u = I(u) \quad \text{or} \]

\[ (T T^{-1})(v) = T(T^{-1}(v)) = T(u) = v = I(v) \]

The identity transformation is a mapping from a state onto the state itself and is effectively an empty transformation.

In other words, the pattern transformations that define a pattern system form a Transformation Group. A group is a set \( G \) that together with a law of composition satisfies the following four axioms:

1. Closure. For all \( a, b \in G \), the set \( G \) is closed under composition: \( ab, ba \in G \).
2. Associativity. For all \( a, b, c \in G \), the composition is associative: \( (ab)c = a(bc) \).
3. Existence of an Identity Element. For all \( a \in G \), there exists an element \( e \in G \) such that \( ae = a = ea \).

4. Existence of Inverses. For each \( a \in G \), there exists an \( a^{-1} \in G \) such that \( aa^{-1} = e = a^{-1}a \).

If we can define a Transformation Group for a pattern, then we can argue that if \( p, \bar{p} \in C \) and \( p \neq \bar{p} \), then for them to be representatives of the same class \( C \), there must exist some transformation, or set of transformation, \( T \), such that \( T(p) = \bar{p} \) where \( T = \{T_1, \ldots, T_m\} \). Therefore, for us to recognise \( q \) all we need to do is find the transformations \( T \), where \( T(p) = q \), then we can say \( q \in C \). The problem is how do we determine the transformations \( T \)? For the moment we will assume we know a set \( \bar{T} \subset T \), which forms the primitive basis of all \( T \). Then, we can say \( \bar{T}(q) = p \), where \( \bar{T} = \{T_1, \ldots, T_n\} \). The problem now is how to determine which combination of transformations, \( \bar{T} \) are applicable to the transformation of \( q \) so that we can determine whether \( q \in C \). There could be a number of transformations all acting collectively upon \( q \) such that to determine the transformation \( \bar{T}_i \) is difficult, especially when \( q \) is not a member of \( C \). Consequently we need to optimise the selection \( \bar{T}_i \), so that we can determine, within a finite number of steps, \( q \)'s class.

Optimising the selection of transformations can be achieved by classifying \( q \)'s present state/condition. If we know what transformation state \( q \) is in, then we can consequently apply the inverse of that transformation to determine whether \( q \) is an element of \( C \). However, to classify all possible states of \( q \) may be difficult. The present state of \( q \) may be a consequence of the effect of many transformations acting
on \( q \) to varying extents. It is even possible that some of these states may not be readily identifiable as they could be occluded by the effect of other transformations. What we need to do is find the dominant transformation, that is, the transformation that has the most significant effect on \( q \). For example, in an image recognition problem, poor lighting conditions might prevent the recognition of other transformation states, i.e. the effect of the lighting dominates the effects of other possible transformations. For recognition too proceed we need to reverse the affect of the dominant transformation, poor lighting and allow the other transformations to be identified. We can then progressively address the affects of each successive dominant transformation until no more transformation states on \( q \) can be identified. We can then determine whether \( q \) is an element of \( C \).

Let \( c = \{c_1, \ldots, c_m\} \) where \( c \) represents the set of transformation state classifiers, then

\[
c_j(q_{r-n}) \xrightarrow{T} c_j(q_{r-n+1}) \xrightarrow{T_m} c_j(q_{r-n+2}), \ldots, c_j(q_r) \xrightarrow{T_m} p \in C
\]

The dominant state can be determined by reviewing the responses from the various classifiers, such that

\[
d(c_1(q_{r-n}), \ldots, c_m(q_{r-n})) \xrightarrow{T} d(c_1(q_{r-n+1}), \ldots, c_m(q_{r-n+1})) \xrightarrow{T_m} d(c_1(q_r), \ldots, c_m(q_r)) \xrightarrow{T_m} p \in C
\]

As \( T \) may not be perfect, e.g. rotating an image will potentially introduce digitization and quantization errors, we need some tolerance with the classifiers to make allowances for considerable variation. We can represent this with the following

\[
c_j(q_{r-n}) > \alpha
\]

and

\[
d(c_1(q_{r-n}), \ldots, c_m(q_{r-n})) > \beta
\]

where \( \alpha \) and \( \beta \) are a arbitrary thresholds.
However, this introduces a significant problem. By recognizing that $T$ may not be prefect and therefore allowing our classifiers to be more flexible we introduce the possibility that we may not be able determine when $q$ has reached its optimum class state. $q$ may be so degraded that it cannot be identified as an element of $C$. Alternatively, the transformation process might continue indefinitely as the data degradation caused by $T$ creates the illusion of different states.

To resolve these issues we need to extend our model by taking into consideration what is involved in the recognition process - what stages are evident and what attributes are needed. Therefore, let us consider a simple example; imagine walking down a dark, secluded alley, and being distracted by the sound, or sight, of a large, moving object. After observing the general area of the disturbance, we begin to perceive that the moving object is a scavenging animal. For some unknown reason, possibly anxiety, this animal launches itself at us - a very excited Doberman, seemingly intent on rending our flesh with its powerful jaws and razor sharp teeth. It is only as the dog is in mid-flight, that we recognise our friend's friendly, placid dog, Fred. What this example endeavours to illustrate is the steps and attributes required to recognise a particular data pattern. Initially we were only aware of a moving object, this assessment was rapidly changed and the object became an animal, then a dog, a specific dog and finally a known dog. We can summarise this process as follows:

1. An event, e.g. an interesting object that attracts attention, or causes a focusing of attention
   - Large moving object.
2. A means of extracting information
   - Automatically accommodating vision system.
3. A means of placing this information in context
• Object is scavenging.
• Animal is running.
• Dog is snarling.

4. Knowledge of different data properties.
   • Knowledge of scavenging characteristics enables transformation of object into animal.
   • Knowledge of dogs allows the transformation of snarling animal into snarling dog and eventually into non-snarling, friendly, placid dog.

5. A means of evaluating the data’s similarity with known data classes.
   • Comparison of placid dog with known dogs, allows association to be made with Fred.

Each of these steps and attributes were used in the example in a series of classification cycles to identify an initially unknown object. The seamless interaction of each component creates an illusion of a conceptual whole that gives the impression that recognition is a singular process. However, it is unlikely that a simple classifier or even a series of classifiers could easily accomplish this task. For example, if we ignore the visual tracking of the object, and simply sub-sample the data, we are left with a series of still images of a moving dog. These images would contain considerable head, body and leg movements, consisting of a variety of different textures and colour that would vary with different dogs. This would make the training and development of a classifier(s) difficult - the data having too many variations for a system to possibly accommodate. However, if we simplify the task and provide the system with knowledge of dog dynamics, it can then relate the sub-sample data instances to this knowledge, enabling allowances to be made for these states. This extra level of abstraction shifts the emphasis away from data recognition to the consideration of the data’s condition/state. This information can then be used to allow the actual recognition of the pattern data to proceed with a relatively, consistent,
normalised data instance. The effect of this normalisation is to reduce the learning and search requirements to more realistic levels.

We can define the required components as follows, we need:

i) an array of elements that define the data pattern, \( P \).

ii) a series of feature extraction routines, \( F \).

iii) a collection of classifiers, \( C \), that define different \( P \) states.

iv) a decision-making apparatus, \( D \), that selects different courses of action.

v) a series of transformations, \( K \), that define the data pattern's dynamics.

vi) a recognition mechanism, \( R \), for determining the class of a data pattern.

vii) a database of class associations, \( CL \), that enable the naming of different classes.

With this basis, if we let \( p \in P \), and transform \( p \) using an isomorphic transformation \( k \in K \), that is limited to the physical dimensions of \( P \), we create a new data pattern \( q \), that is derived from \( p \), such that \( p \neq q \). Although we have changed the original pattern, \( p, q \) is still an element of \( P \). This transformation has changed \( p \)'s representation but has not affected the intrinsic pattern that \( p \) represents. Therefore, if we reverse the transformation on \( q \), \( k^{-1}(q) \), and create \( r \), then \( p \) will equal \( r \).

We can therefore argue that a pattern is defined by the way in which the pattern data interacts with itself, not by the way that its supporting structure presents it. This means that a data pattern is independent of its supporting structure. However, whether we can associate the observed data pattern to a known pattern is limited by our
capacity to reverse the effects of the transformations that have acted on the observed pattern.

Let $p, q \in P$ and let $N(p)$ denote the set of patterns generated by $p$ and let $c(q)$ denote $q$'s class. Then $c(q) \cap N(p)$ is either empty or equal to $c(q)$. From this it follows that $N(p) = \cup c(q)$ for some set $T \subset P$.

Now let $N_A$ denote the number of elements of the set of $A$. For $p \in P$, let $\|N(p)\|$ denote the number of equivalent classes in $N(p)$, that is,

$$\|N(p)\| = \text{Min}\{N_T : T \subset P, N(p) \cup c(q)\}$$

$\|N(p)\|$ is equal to the number of distinct classes it contains, from which it follows that $\|N(p)\| \geq 1$ for all $p \in P$. If $X$ is a pattern, then all patterns that can be derived from $X$ are the same as $X$ as long as we can model the transformations applied to $X$, consequently $\|N(X)\| = 1$.

This is an important conclusion as it states that from a single pattern instance all other pattern instances can be determined, within the physical constraints of the original pattern instance. This reaffirms the argument that a pattern is still the same pattern irregardless of the changes made to that pattern's underlying structure. An exception to this would be non-isomorphic transformations that change the pattern's inherent structure by tearing or folding. These types of transformation may be difficult and in some cases impossible to reverse as the tearing or folding causes information to be lost and data relationships destroyed.
We can conclude from this that as long as we can reverse the effect of a pattern transformation then we can eventually find an example of that pattern that will be consistent among all instances of that pattern.

The difficulty is in determining the transformations required to associate the novel pattern instance with a known pattern. To accomplish this we need to be able to classify the pattern’s present state so that a suitable transformation can be made that increases the similarity between the known and unknown pattern instances. As there may be more than one transformation affecting the pattern instance, it may be necessary to transform the pattern several times. An exception to this would be if the entire pattern space was already known. In this situation it would not be necessary to perform any transformations as the known pattern instances would already define the novel pattern instance.

3.3 Dynamic Recognition Neural Network Model

The basic premise of the DRNNM is that recognition is easier if the data is in a known state or condition. In this way, different pattern instances can be easily compared as extraneous factors that may inhibit recognition can be removed, or minimised. However, for the data to be normalised the recognition process must be dynamic. A static recognition system can provide mappings and associations that minimise the effects of different data states but it cannot change the data’s state. Therefore, a static system cannot remove the effects of different states upon the pattern data, and consequently any training bias may significantly affect recognition.

The DRNNM changes a data pattern’s state by forcing the initial data pattern to undergo a series of transformations until it, ideally, reaches a final recognisable state. This final state is often referred to as a data attractor, as it is the data state towards
which the data is transformed. When data is presented to the system it can be viewed as representing a point on an attractor landscape, with the DRNNM providing the energy to move across and around this area. The attractor can be visualised as a large basin, with the different DRNNM transformations acting as gullies, feeding into this basin.

Recognition systems are not limited to single data attractors. Using this analogy with simple static recognition systems the attractor becomes equivalent to a class prototype or exemplar. In these systems recognition is based purely on the nearness of the data's point representation to a specific attractor, e.g. Nearest Neighbour. Consequently, the more points there exist to characterise a class, the greater the chance of accurately recognising that class. However, in situations where there exist limited numbers of attractors, as when training systems with minimal examples, it is possible for a class instance to be a long way away from the appropriate attractor. This presents a problem to static systems as it has no way of moving the point to more accurately verify the data's identity.

The main advantage that the DRNNM has over static systems is its ability to change the data's state. However, the transformation of the data from an initial unrecognisable state to a recognisable state is not a simple procedure. There has to exist some means of determining what state the data is in and a means of transforming that data within the constraints of the data's domain. It is not acceptable to simply change the data so that it represents a known class as the initial data pattern being recognised will no longer exist. This process is addressed in more detail below.

We can abstract the recognition process provided by the DRNNM with the following operator function:

Let DRNN be the model operator.
Let $C = \{C_1, C_2, C_3, \ldots, C_n\}$ be the possible recognition classes, and let $O = \{O_1, O_2, O_3, \ldots, O_n\}$ be the dynamic data set.

**DRNN**: $O \rightarrow C$

$$O \xrightarrow{\text{DRNN OperatorSet}} C$$

The DRNN operator set's function is to assign the pattern data, $O$, to a recognition class, $C$. This differs from other recognition approaches in that we do not try to assign a recognition class to the pattern data. The operator's objective is to transform the pattern data, if necessary, to determine whether it belongs to a recognition class. It is possible that the pattern data is not classifiable and has no applicable attractor/class representation. This allows our recognition system to be less flexible as it does not have to concern itself with recognizing sub-optimal data instances. If the data is not in a suitable state to be recognized the DRNN will not try and recognize it. It will try and determine whether the data can be transformed into a recognizable state. This eliminates problems associated with trying to recognize data when only minimal, or limited, data class examples exist. We no longer have to worry about whether the system will generalize sufficiently to cater for novel instances or whether that generalization capability will result in data being incorrectly classified as something else - false positives.

To enable the DRNN operator to provide this functionality it consists of a number of sub-operators that represent different aspects of the transformation and recognition process.

We can define the DRNN as follows:

$$\text{DRNN} = \{\text{PASO, FSO, CSO, DSO, KSO, RSO, CLO}\}$$
where

\[ \textit{PASO} = \text{Pattern Activity Space Operator.} \]
\[ \textit{FSO} = \text{Feature Space Operator.} \]
\[ \textit{CSO} = \text{Classifier Space Operator.} \]
\[ \textit{DSO} = \text{Decision Space Operator.} \]
\[ \textit{KSO} = \text{Knowledge Space Operator.} \]
\[ \textit{RSO} = \text{Recognition Space Operator.} \]
\[ \textit{CLO} = \text{Classifier Label Operator.} \]

These sub-operators interact to provide the necessary functionality needed to allow the data recognition process the capability to accommodate the variability of dynamic data. The operators create a domain specific environment where a recognition problem can be artificially reassociated with its environment. Figure 3.1 provides an overview of how each operator space interacts and communicates with each other operator.
Figure 3.1 Dynamic Recognition Neural Network Model

Each of the operators is further defined and explained in the following sections.
3.4 DRNNM Sub-Operators

3.4.1 Pattern Activity Space

3.4.1.1 Introduction

Pattern Activity Space provides the interface between the real world and the DRNNM. It is concerned with the underlying data independent structure and predefined physical dimensions that support the recognition data. In effect it is a representation of all the possible pattern instances, within a recognition domain, at a particular point in time. We can describe its underlying structure as follows:

If a set $S \subset PS$, where $PS$ denotes all possible pattern states, then $S$ is defined by

$$S = \{(x_1, x_2, ..., x_n) \in PS : a \leq x_1 \leq b, c \leq x_2 \leq d, ..., e \leq x_n \leq f\}$$

where $a < b$, $c < d$ and $e < f$ are real constants, that define $PS$'s physical dimension to $(b - a)$ units, $(d - c)$ units and so forth. The units are the measurement units used to obtain the pattern data, for examples, pixels.

We can say that the set $S \subset PS$ gives the extent of the pattern. It defines the limit, or boundary, of acceptable pattern instances within the model. If there were no data limitations, or data structure, then comparing different data instances would be more difficult as there would be no common basis to establish similarity.

3.4.1.2 Pattern Activity States

Let $PAS = \{PS_t - i, PS_t - 2, ..., PS_t - v\}$ then we can define two general Pattern Activity States.

1. Initial - represents the starting state
\[ O, NIL \xrightarrow{\text{PASO}} \text{PAS} \text{ where} \]

\( O \) = data to be recognized and \( \text{NIL} \) implies the initial state, no transformation(s) have been applied.

At this stage the data has just been received and has not undergone any processing, other than ensuring that it is within acceptable parameters. For example, raw grayscale image data with values ranging from 0 – 255 might be mapped to values ranging from 0 – 1.

2. Normal - the state after each successive recognition/ transformation cycle.

\[ \text{PAS, KS} \xrightarrow{\text{PASO}} \text{PAS} \text{ where} \]

\( \text{KS} \) = Knowledge Space - discussed later.

\( \text{PASO} = \{\text{PASSO, PAMSO}\} \) where

\( \text{PASSO} = \) Pattern Activity Standard Space Operator.

\( \text{PAMSO} = \) Pattern Activity Memory Space Operator.

The DRNN process is a dynamic, cyclic process. Data cycles through the DRNN until the transformation process is completed and the data is considered recognizable or not recognizable. How the transformations are performed is controlled by the Knowledge Space Operator, \( KSO \), based on instructions from the Decision Space Operator, \( DSO \).

The product of each transformation is the next PAS instance that the DRNN processes.
3.4.1.3 Pattern Activity Space Operators

As the DRNN is a cyclic process we have the option of remembering previous PAS states by merging them with the present PAS. This has the potential to generate a similar affect to that provided by Optical Flow. Why would we want to do this? Whenever data is transformed by the KSO the state of the previous data is potentially lost. By integrating the previous data with existing data some of this information will be maintained and may simplify subsequent classification of the data’s state. For example, if all you are shown is single frames of a tennis ball in mid-air it may be difficult to determine the ball’s velocity. However, if an after image is present that gives an indication of the ball’s previous position it becomes easier.

We can define two operators that capture this concept:

1. Pattern Activity Memory Space Operator – PAMSO

If we define Pattern Activity as the effect of the pattern recognition data on Pattern Activity Space then we can define the PAMSO as follows:

\[
PAS_{t-n}, \ldots, PAS_{t-3}, PAS_{t-2}, PAS_{t-1} \xrightarrow{\text{PAMSO}} PAS_t
\]

The PAMSO provides a gradual decrease in Pattern Activity over a period of time.


The PASSO differs from the PAMSO in that it reinitialises Pattern Activity Space after each recognition cycle.

\[
PAS_{t-1} \xrightarrow{\text{PASSO}} PAS_t
\]
These concepts are illustrated in the following diagrams

**Figure 3.2 Representation of Pattern Activity Space after PASSO.**

**Figure 3.3 Representation of Pattern Activity Space after successive PAMSO.**

The *PAMSO* establishes a link between pattern activities after successive transformations. As mentioned previously this is a similar concept to Optical Flow in vision research - the incoming data causes increased neural activity which takes a period of time, \( \alpha \), to return to its original state (*Time Dependent Neural Activity*). This reduces rapid transitions, or non-linearities, between different pattern states as the transient activity ensures that previous states are still partially active. This is
illustrated in Figure 3.2 and Figure 3.3, where the correlation between the final and the previous state \((t-1)\) in Figure 3.2 is substantially lower than in Figure 3.3.

### 3.4.2 Feature Space

#### 3.4.2.1 Introduction

The objective of Feature Space is to allow the DRNN to utilise different feature types when and if required. A study by Jones et al. (1991) found that young children (2-3 years) interpreted objects with and without eyes in fundamentally different ways. They used shape and texture when viewing objects with eyes and only shape when viewing the same objects without eyes. This suggests that even at a young age children select different feature sets under different contexts. To accommodate this insight the DRNN does not restrict itself to a limited feature set as this might prevent the DRNN from appreciating the effects that different contexts may have on the data it receives.

We can define Feature Space as a collection of different feature extraction processes through which the DRNN can access different feature representations of the data and thereby obtain different perspectives of the data. Why is this important? In face recognition, for example, some of the features sets used include grayscale data [Turk], edge data [Nakamura 89 & 91], local defined features - such as eyes [Craw 92][Yuille][Sakai] or undefined local features [Malsburg 92][Crowley 94d] and even laser range finding data [Lee]. Which of these features sets is the most effective is dependent upon the conditions in which they are applied. For example, in face recognition, it is generally accepted that coarse, full facial recognition techniques will handle localized changes caused by glasses, or facial hair, better than techniques dependent upon specific facial features. However, systems using specific features are often more effective at handling global facial changes, such as slight scale changes or
rotations. So by having a variety of feature sets available allows the potential
development of a more robust, recognition system.

An important aspect of Feature Space is that it separates the feature extraction process
from the recognition process allowing the evaluation of different feature sets under
different conditions and the selection of the most applicable feature set for specific
conditions. It also enables the addition of new feature sets without affecting the
existing system. This differs from traditional recognition systems where the
recognition process and the features that its uses are tightly coupled. Therefore
incorporating new features or different recognition strategies can be very difficult.

3.4.2.2 Feature Space Operators

We can define Feature Space as follows:

Let

\[ FS = \{ FS_1, FS_2, FS_3, \ldots, FS_n \} \]

be the set of Feature Spaces where

\[
\left\{ FS_1, FS_2, FS_3, \ldots, FS_n \right\} = \begin{bmatrix}
fs_{11} & \cdots & fs_{1n} \\
\vdots & \ddots & \vdots \\
fs_{m1} & \cdots & fs_{mn}
\end{bmatrix}
\]

are the feature extraction processes that each Feature Sub-Space contains.

Then:

\[ PAS \xrightarrow{FSO} FS \]

where

\[ FSO = \{ FSO_1, FSO_2, FSO_3, \ldots, FSO_n \} \]

is the subset of Feature Space Operators and

\[ PAS_n \xrightarrow{FSO} FS_m \]
Therefore one or more, feature sets may be extracted from the data pattern. For face recognition, we might have a feature extraction process that extracts the position of specific features such as a person's eyes. And another process for extracting line data, and another for textural information. The DRNN may only use one of these feature sets, or it may use all of them.

3.4.3 Classifier Space

3.4.3.1 Introduction

The objective of Classifier Space, CS, is to provide general information about the state of a pattern so as to allow the effect of this state to be taken into consideration. As the data is unknown and static, the system cannot interact with the mechanism used for obtaining that data, e.g. the system cannot take another picture to clarify a critical area, or obtain a different perspective – the image capture device is presumed to be outside of the DRNN's control. Therefore, CS is used to give some indication of what and where the recognition specific pattern data is, so that recognition can proceed. Without CS there is no means of determining what transformations are required to normalise a data sample.

After the data has being processed by the Feature Space Operator the feature sets generated are analysed by the set of classifiers that comprise CS and based on their responses a decision is made by the Decision Space Operator concerning the presence of different pattern states.
3.4.2.2 Classifier Space Operators

We can define the relationship between Feature Space and Classifier Space as follows:

\[ FS \xrightarrow{CSO} CS \]

As there may be a number of different features sets available in Feature Space we need to have a different set of classifiers for each feature set, therefore we need to decompose \( CSO \) into a series of sub operators, therefore

\[ CSO = \{ CSO_1, CSO_2, CSO_3, ..., CSO_n \} \]

where

\[ FS_1 \xrightarrow{CSO} CS_1 \]
\[ FS_2 \xrightarrow{CSO} CS_2 \]
\[ FS_3 \xrightarrow{CSO} CS_3 \]
\[ \vdots \]
\[ FS_n \xrightarrow{CSO} CS_n \]

Each feature set \( FS_x \) requires its own set of classifiers, with each set of classifiers having its own series of sub-operators responsible for classifying different states.

\[
\{ CSO_1, CSO_2, CSO_3, ..., CSO_n \} = \begin{pmatrix}
CSCO_{11} & \cdots & CSCO_{1n} \\
\vdots & \ddots & \vdots \\
CSCO_{m1} & \cdots & CSCO_{mn}
\end{pmatrix}
\]

Therefore for each different FS there exists a corresponding CS that contains a number of specific classifiers appropriate for the recognition of different data states using the applicable feature set.
So what does the classification process generate in response to the data it receives? This depends on the classifier’s role. We may want to only know whether the data is in a particular state, e.g. an unrecognizable or recognizable state. Alternatively, we may want an indication of to what extent the data is in a specific state. Or we may just want a relative indication based on the effects of opposing states, e.g. more left than right. To provide this information we need to extend our initial definition of Classifier Space to include the following sub-spaces:

$$CS = \{BCS, CCS, ACS\}$$

where

- **BCS** = Binary Classifier Space.
- **CCS** = Continuous Classifier Space.
- **ACS** = Associative Classifier Space.

Each of these sub-spaces represents different classification approaches to determine different types of pattern states and contain further sub-spaces that define specific classifications.

$$BCS = \{BCS_1, BCS_2, BCS_3, \ldots, BCS_n\}$$

$$CCS = \{CCS_1, CCS_2, CCS_3, \ldots, CCS_n\}$$

$$ACS = \{AGCS_n, AGCS_p\}$$

where

- **AGCS_n** = Associative Group Classifier Space Negative
- **AGCS_p** = Associative Group Classifier Space Positive

and

$$AGCS_n = \{AGCS_{n1}, AGCS_{n2}, AGCS_{n3}, \ldots, AGCS_{nm}\}$$

$$AGCS_p = \{AGCS_{p1}, AGCS_{p2}, AGCS_{p3}, \ldots, AGCS_{pm}\}$$
3.4.2.3 Classifier Subspace Operators

These sub-spaces allow us to define several sub-operators:

1. Binary Classifier Space Operator

\[ FS \xrightarrow{BCSO} BCS \]

where

\[ BCS = \{0, 1\} \]

BCS is concerned with the binary classification of FS. The BCSO maps FS to TRUE or FALSE values. This is used to classify a state as being present or absent in the data pattern.

2. Continuous Classifier Space Operator

\[ FS \xrightarrow{CCSO} CCS \]

where

\[ CCS = [0, ..., 1] \]

The CCS classifies FS with a value between 0 and 1; The CCSO mapping FS to a value that is indicative of the extent of a particular state in the pattern data. This operator might be used to define the distance of a point from the origin, or the relative size of an object.

3. Associative Classifier Space Operator

\[ FS \xrightarrow{ACSO} ACS \]

where
\[ \text{if } (AGCS_{nx} > \alpha) \& (AGCS_{px} < \alpha) \& (|AGCS_{nx} - AGCS_{px}| > \beta) \]
\[ \text{then } \]
\[ AGCS_{nx} = |AGCS_{nx} - AGCS_{px}| \]
\[ \text{else } \]
\[ ACS_x = 0 \]

or

\[ \text{if } (AGCS_{nx} < \alpha) \& (AGCS_{px} > \alpha) \& (|AGCS_{nx} - AGCS_{px}| > \beta) \]
\[ \text{then } \]
\[ AGCS_{px} = |AGCS_{nx} - AGCS_{px}| \]
\[ \text{else } \]
\[ ACS_x = 0 \]

The ACS is more complex than the other sub-spaces and is used to indicate the relative variation between two opposing states. Consequently, the value that the ACSO maps $FS$ to is dependent upon the difference between opposing states. This classification is applicable to situations where there exists a relationship between two states. For example, we might classify an object, as in the example below, as being more left than right.
After we have classified the state of the data we have to determine what to do with it. This is what Decision Space is for.

3.4.4 Decision Space

3.4.4.1 Introduction

Decision Space is concerned with determining an appropriate action based on past and present classifier information. Classifier Space provides Decision Space with information about the present state of the data. When this information is used in conjunction with previous Classifier data then different system behaviours/characteristics can be defined.

Why do we use both past and present classification information? If we only use present classifier information there is a risk that the DRNN will oscillate between two or more different states. That is, a decision based on the data’s present state results in

Figure 3.4 Relative Classifier Responses to differently positioned data
a transformation that pushes the data into an opposing state that results in the classifiers responding to the new state and causing the reverse transformation, thereby creating a continuous, never-ending cycle. By using both past and present classification information it becomes possible to accommodate and classify DRNN transformation behaviour, thereby avoiding potential processing problems.

3.4.4.2 Decision Space Behavioural Patterns

We can define some DRNN behavioural patterns as follows:

1. Stable
   a. Positive (recognisable)/Excited - the data is recognisable. This behaviour is indicated by a very positive CSO response over a series of transformation cycles. The data is rapidly transformed into a recognizable state.
   b. Negative(unrecognisable)/Bored - the data is unrecognizable. The CSO are not responding to the data.

2. Unstable/Chaotic/Confused - the CSO responses vary abruptly after each recognition cycle. Once the data’s state has been classified it should be possible to then systematically transform the data into a recognizable state. This behaviour suggests that the data is not recognizable or that the classifiers require further training.

3. Oscillating/Cycling/Anxious - the same CSO responses are occurring every few recognition cycles.
4. Exploratory/Curious - the CSO responses are improving with each recognition cycle.

### 3.4.4.3 Decision Space Learning and Operators

Learning is dependent upon the dominant system behaviour over one or more recognition cycles or sessions.

Let

\[ CS \xrightarrow{DSO} DS \]

then we can define two learning sub-spaces

\[ DS = \{SDS, LDS\} \]

- **SDS** = Short Term Learning Decision Space
- **LDS** = Long Term Learning Decision Space

that represent Decision Space subsets responsible for learning. Short Term learning is learning that occurs within a recognition session, whereas Long Term learning refers to learning that occurs over multiple recognition sessions. Both Short and Long Term learning provide a means of streamlining the decision making process, by establishing persistent associations between Classifier Space and Decision Space. The classifier information is associated with specific Decision Subspaces that represent the possible decisions to the classifier response. We can represent these Decision Subspaces as follows:

\[ \{DS\} = \{RDS, KDS, TDS\} \]

where

- **RDS** = Recognition Decision Subspace.
Each of these subspaces represents a course of action by Decision Space. The selection of RDS means that the data is in a normalized state and should be recognisable; whereas KDS indicates that the data is not recognisable and needs further adjustment and the selection of TDS is indicative that the data will never be recognisable and the recognition process should terminate.

This allows us to define Decision Space operators and sub-operators

\[ DSO = \{ DSO_t, DSO_s, DSO_a \} \]

where

- \( DSO_t \) = Decision Learning Operator.
- \( DSO_s \) = Decision State Operator.
- \( DSO_a \) = Decision Activation Operator.

These are described in more detail below:

### 3.4.4.3.1 Decision Learning Operator

This operator is responsible for maintaining and developing associations between classifier responses and system decisions. We can define two sub-operators as follows:

Let \( DSO_t = \{ SDSO, LDSO \} \)

where

- \( SDSO \) = Short Term Learning Decision Space Operator
\( LDSO = \) Long Term Learning Decision Space Operator

1. Short Term Learning Operator

\[ DS \xrightarrow{SDSO} SDS \]

Within a single recognition session the data’s state may be classified many times and subsequent decisions made. This operator maintains information regarding this link between classifier response and associated action. This enables Decision Space behavioural patterns to be recognised and accommodated within a recognition session.

2. Long Term Learning Operator

\[ DS \xrightarrow{LDSO} LDS \]

Over many recognition sessions certain classifier responses will consistently be associated with specific system responses. The objective of this operator is to maintain and strengthen these associations, thereby enabling the system to recognise possible transformation strategies.

3.4.4.3.2 Decision State Operator

This operator, in conjunction with \( DSO \), determines the present behavioural state the system is in. The different states can be represented as follows:

\[ DSO = \{ DSSO, DSUO, DSOO, DSEO \} \]

where

\( DSSO = \) Decision Space Stable Operator

\( DSUO = \) Decision Space Unstable Operator

\( DSUO = \) Decision Space Oscillating Operator

\( DSEO = \) Decision Space Exploratory Operator

1. Stable State
CS \xrightarrow{DSSO} RDS

where

\textbf{DSSO:}

\begin{align*}
\text{if } (CS > \alpha) \\
\text{then } DS = RDS \\
\text{elseif } (CS < \beta) \\
\text{then } DS = TDS
\end{align*}

with \( \alpha \) representing an arbitrary threshold signifying a high positive classifier response and \( \beta \) representing a very low negative classifier response.

2. Unstable State

\( CS \xrightarrow{DSUO} TDS \)

where

\textbf{DSUO:}

\begin{align*}
\text{if } (CS_{r-z} > \alpha) \& (CS_{r-y} < \beta) \& \ldots \& (CS_{r-n} < \alpha) \\
\text{then } DS = TDS
\end{align*}

with \( \alpha \) and \( \beta \) being arbitrary thresholds that indicate acceptable high and low classifier responses. \( (CS_{r-z} > \alpha) \& (CS_{r-y} < \beta) \& \ldots \& (CS_{r-n} < \alpha) \) refers to successive classifier responses over a defined period of time, i.e. a number of classification cycles. How many classifier cycles are required before a decision can be made depends on the domain, or it could be based on empirical data. It may also not involve all classifier responses but a subset of classifiers.

3. Oscillating State

\( CS \xrightarrow{DSCO} TDS \)

where

\textbf{DSCO:}
The classifier responses over a number of cycles continually repeat themselves. This does not necessarily apply to all classifier responses, a subset of the classifier responses may be sufficient for this state to be recognised. In general, if the data transformation process introduces any artifacts into the data it is unlikely that this state will be observed.

4. Exploratory State

\[ CS \xrightarrow{DSEO} KDS \]

where

\[ DSEO: \]

\[
\text{if } \left( \left( CS_{r-n} < CS_{r-(n+1)} \right) \& \left( CS_{r-(n+2)} < CS_{r-(n+1)} \right) \& \ldots \& \left( CS_{r} < CS_{r-n} \right) \right)
\]

\[ \text{then } DS = KDS \]

\[ \text{endif} \]

If the classifier responses during a recognition session, or over a number of classification cycles, are consistently improving than the system is in an Exploratory State.

3.4.4.3.3 Decision Activation Operator

These operators define the different states the system is in together with the Decision Space preference.

We can define the activation of either RDS, KDS and TDS as follows:
Let \( DSO_s = \{DKSO, DRSO, DTSO\} \) where

\[ \begin{align*}
DKSO &= \text{Decision Knowledge Space Operator} \\
DRSO &= \text{Decision Recognition Space Operator} \\
DTSO &= \text{Decision Termination Space Operator}
\end{align*} \]

1. Activate Recognition Decision Space

\[ DS \xrightarrow{DRSO} RDS \]

This operator is responsible for forwarding the data to Recognition Space.

2. Activate Knowledge Decision Space

\[ DS \xrightarrow{DKSO} KDS \]

DKSO indicates the appropriate data transformation to be performed by Knowledge Space.

3. Activate Termination Decision Space

\[ DS \xrightarrow{DTSO} TDS \]

DTSO terminates the recognition session.
3.4.5 Knowledge Space

3.4.5.1 Introduction

The objective of Knowledge Space is to provide access to data transformations that will allow evaluation of the recognition data from different perspectives without affecting the data’s structural integrity. This separates the pattern dynamics from the recognition process; thereby permitting the system to concentrate on recognising the pattern, rather than on trying to:

- find associations between different data representations, of what is intrinsically the same pattern.
- weight different data locations to improve the discriminatory capabilities of the extracted features.
This restricts Knowledge Space to transformations that change the data's state, ensuring that the system's knowledge is clearly defined and the system's capabilities easily determined.

In a more abstract sense Knowledge Space defines the dynamics of the pattern domain. It restricts the possible pattern variability to reasonable limits. The transformations that it provides are all feasible within the pattern domain and tie the data transformations to that domain. We, for example, do not generally worry about possible events outside our experienced laws of physics - objects move and interact in well defined ways that we learn at a very young age [Kellman]. However, this does not mean that contained within our vision system is the capability to appreciate the affects of all possible transformations on all objects within our immediate area. If we see something of interest we move our head and eyes and fixate our visual system on that area of interest. We have learnt from early childhood that by physically adjusting out visual system there is a better chance that we will be able to determine what is there. We do not generally try and determine the nature of something with our peripheral vision. We focus on it. Knowledge Space provides the mechanisms required to allow the DRNN the ability to adjust its perspective and focus on the area of interest. This capability is not learnt by our visual system; it is provided by our physiology and utilized by the visual system. Although it may take time to learn coordinate both systems.

### 3.4.5.2 Knowledge Space Operator

The Knowledge Space Operator, \( KSO \), selects the transformation specified by Decision Space.
We can define $KSO$ as follows:

Let $DS \xrightarrow{KSO} KS$

where

$$KSO = \{KSO_1, KSO_2, KSO_3, \ldots, KSO_n\}$$

and

$$KSO_1 = \text{Transformation Type 1}.$$  
$$KSO_2 = \text{Transformation Type 2}.$$  
$$KSO_3 = \text{Transformation Type 3}.$$  
$$\vdots$$  
$$KSO_n = \text{Transformation Type n}.$$  

The type and number of transformations available depends on the data’s domain. The Knowledge Space Operator applies the transformation to the data in PAS. This creates a new data instance which is pushed through the DRNN process again.

### 3.4.5.2 Explanation

The concept of knowledge in this context is a significant change from traditional classification definitions of knowledge. In sub-symbolic recognition systems, knowledge is usually referred to as the features encoded in the structure of the recognition system during training. Knowledge is perceived as being the feature associations the system develops to enable discrimination among the different training classes. These features define the class, enabling its accurate classification under non-ideal or novel conditions. However, we argue that the system has no knowledge of the class or the data domain it represents; it only knows how to associate different data locations with a particular class. It does not matter how many layers, the type of
activation function or the learning rules that are used, the system will only ever learn how to map one piece of data with another. Even if such an elaborate system was developed, it would have no understanding of where the pattern, e.g. a character, or face, is contained within the data – the context would be lost.

Traditionally in sub-symbolic systems knowledge and feature associations are considered as virtually equivalent. However, this is not strictly accurate. Features are essentially nothing more than abstract interpretations of the data, and therefore provide no real knowledge of the behaviour or qualities of the data. Features only give an indication of the existence of particular data traits or characteristics. Knowledge though should provide an understanding of how the data interacts and is affected by the environment. It should be capable of transforming the data without affecting the data’s underlying structure.

Features only allow data to be interpreted. The more features, and the more complex the associations between those features, the more complex the features the system can develop and interpret. But this does not mean that the system has any knowledge of what it is interpreting. For example, neural networks are often used in an attempt to "model" a pattern’s dynamics through training examples. However, all that the model has learnt is a complex feature association. It has learnt that a particular combination of data provides a particular response. Therefore when a pattern instance is presented to the system, it simply extracts and weights different data regions and determines what spatial region that pattern corresponds to. A model developed in this fashion has no appreciation of why patterns change, or what causes these changes. It does not
necessarily learn any suitable data associations. It learns whatever associations result in an adequate solution that satisfies the immediate objective.

Nevertheless, the failure of these systems is still deemed to be a problem with the recognition model or the training examples. Dubious explanations, such as: "The system didn't find the global minima", "over-training/over-fitting", "not enough feature vectors or hidden units", "inappropriate training examples", "it didn't learn what it was supposed to learn", abound in the recognition literature for the failure of recognition systems to acquire the desired knowledge and understanding of the problem. A more reasonable explanation is that the problems were created by inappropriate use of the classifiers. This is not to suggest that the feature relationships that these systems provide are not worthwhile. Studies have indicated that Neural Networks and other classifiers may be capable of finding better features than their human counterparts [Buchanan].

Knowledge and features are generally separated in symbolic classification systems. But these systems also ignore the difficulty of translating sub-symbolic data into consistent symbolic information. The world does not consist of nice little red pyramids; blue cubes [Winograd] and other easily definable objects. Raw data has to be processed, evaluated and classified, before the functionality of symbolic, rule-based systems can be applied. This separation from the data source prevents effective data interaction and makes these systems less able to handle or adapt to novel situations.
Knowledge Space ensures that the knowledge of the data domain is independent and not confused with the data. This allows the data to be manipulated and changed with minimal loss of data integrity. Consequently a pattern can be recognised independently of its initial state as pattern domain knowledge allows the effect of a particular state to be considered. For example, if a person does not understand that an object can lie on top of another object, then that person will be unable to discriminate either object, as they will perceive the combined object as a single entity [Piaget]. This is because the person is unable to conceive of the object as being independent of the other object. This is irregardless of the features that the person uses to observe the object. However, if the person understands that objects can lie on top of one another then the person is more capable of perceiving that one object is lying upon another object.

### 3.4.6 Recognition Space

Recognition Space provides mechanisms/approaches to recognise the data.

$$\text{DS} \xrightarrow{RSO} \text{RS}$$

The Recognition Space Operator can be any form of recognition and attempts to associate the normalised data with a recognition class.

### 3.4.7 Classification Space

Classification Space provides the class identity/label of the recognised data.

$$\text{RS} \xrightarrow{CLS0} \text{CLS}$$

where
\[ CLS = \{C_1, C_2, C_3, \ldots, C_n\} \]

and

\[ CLSO_i : \]

\[ \text{if} \left( RS \in CLS_i \right) \text{then} \]
\[ \text{result} = C_i \]

\[ \text{else} \]
\[ \text{result} = CLSO_{i+1} \]

### 3.5 Model Operation

We provide here a simple example that describes the basis of how the theoretical spaces interact.

#### 3.5.1 Initialisation

We can model a character or digit recognition problem \( D \) by \( D_i \). This uses only binary information to represent pattern data with any member \( d \) of \( D_i \) being represented by its underlying structure \( S \) - the image plane, and a function limiting the characteristics of \( S \) to the binary values of some measurable subset \( A \subset S \), where \( A \) represents the actual characters found on \( S \), giving it a black and white aspect.

We can define Pattern Activity Space, \( PAS \), as being equivalent to \( S \) except that \( PAS \) is not limited to binary values. In this way time dependent activity can be incorporated.
In this example, Feature Space, FS, acts more as a conduit between PAS and Classifier Space. It does not perform any significant feature extraction. Complex features could be used, such as lines, line-endpoints or line-intersections (in this case this is not desirable).

Classifier Space, CS, is a subset of A, CS ⊆ A, and defines the different states that the members of A exhibit. For character recognition this might consist of the states generated by affine transformations and local topological changes, e.g., scale, rotation and translation. Therefore, CS will consist of a number of classifiers responsible for determining which particular affine, or other, transformation is dominant.

Knowledge Space, KS, is comprised of functions that perform domain specific transformations. In character recognition, this might include limited affine transformations, depending on the system constraints.

Decision Space, DS, initially consists of a novelty threshold that regulates the model’s responsiveness. As the model evolves, Decision Space expands its decision making capabilities to more readily respond to recognition demands and the selection of appropriate responses.

Recognition Space, RS, is comprised of different recognition strategies to evaluate the pattern data.

Classification Label Space, CLS, consists of all the unique members of A, and defines those classes to be used to identify the pattern data.
Learning

The system learns to recognise different states by exploring its artificial environment. When a pattern changes state there is an abrupt increase in the difference between the initial and final state, causing the system to lose stability. This allows an awareness of a pattern's different states to develop, as whenever a transformation causes a significant change between the similarity of the initial and final pattern states, it is indicative of a change in the pattern's overall state. Therefore, by systematically employing its transformation knowledge, a system can learn the effects that different transformations have on the data. It is possible then for a transformation and the associated change to be linked. In this way, the system can develop pathways that link different states. A pattern can then be transformed in a consistent fashion, effectively modeling the data dynamics. The transformation of a pattern into its ideal state is then a simple matter of presenting it to the system and observing at what point the system restabilises.

The evolution of the system can be visualized as a flat surface that develops, through the interaction of Knowledge and Classifier Space, into a series of valleys that lead to a central basin, or catchment area - figure 3.6. This catchment area then branches out into a series of different locations (households) which represent the different recognition classes. The recognition process is thus the equivalent of dropping a drop of water onto this surface and observing where it finally comes to rest. In some cases the water will become stuck somewhere on the transformational surface, the equivalent of a gradient descent local minima. However, unlike gradient descent
models, the existence of a local minima is readily recognisable by, in this case, the water’s location.

Figure 3.6 Normalisation Process Overview

Recognition

The system recognises data in a similar way to how it learns. When data is presented to the system its state is classified and a decision is made to continue processing, terminate processing or attempt recognition. If the system decides to continue processing then an applicable transformation is made to try and maximise the possibility of successful closure in the next recognition cycle. If termination is selected then the recognition process is simply terminated. If recognition is selected then the data class is evaluated and classified.
We can define three basic system operations or states

1. Direct Recognition

\[ O \xrightarrow{\text{PAS}} \text{PAS} \xrightarrow{\text{FSO}} \text{FS} \xrightarrow{\text{CSO}} \text{CS} \xrightarrow{\text{DSO}} \text{DS} \xrightarrow{\text{RSO}} \text{RS} \xrightarrow{\text{CLSO}} \text{CLS} \rightarrow C \]

The recognition data is automatically recognisable and is therefore classified directly. This would be the expected response if the training data was used to test the recognition process.

2. Indirect /Cyclic Recognition

\[ O \xrightarrow{\text{PAS}} \left\{ \begin{array}{c}
\text{PAS} \xrightarrow{\text{FSO}} \text{FS} \xrightarrow{\text{CSO}} \text{CS} \\
\text{KS} \xrightarrow{\text{DSO}} \text{DS} \xrightarrow{\text{RSO}} \text{RS} \xrightarrow{\text{CLSO}} \text{CLS} \rightarrow C
\end{array} \right. \]

In this case the recognition data requires adjustment. Therefore the data is cycled through the system until an appropriate state is achieved.

3. Termination

a. \[ O \xrightarrow{\text{PAS}} \text{PAS} \xrightarrow{\text{FSO}} \text{FS} \xrightarrow{\text{CSO}} \text{CS} \xrightarrow{\text{DSO}} \text{DS} \xrightarrow{\text{CLSO}} \text{CLS} \rightarrow C(\text{Termination Class}) \]

The recognition data is unrecognisable and consequently the process is terminated.
Attempts to improve the quality of the data fail and therefore the recognition process is terminated.

### 3.6 Model Plausibility

Is the DRNN Model approach plausible?

#### 3.6.1 Biological Plausibility

The DRNN Model was not developed to provide insight into biological systems. It was developed to address problems inherent in traditional classification. There was some consideration given to possible biological mechanisms, however this was not the overriding factor in the model’s design. A modular hierarchy was used to simplify the training and development of the recognition system. The recognition process was decomposed into distinct modules to make it easier to address the problems discussed in Chapter Two – by each module having a distinct and well defined function it was proposed that the training of a module would be simplified and problems with a module more readily identified. The DRNNM providing a framework in which all the different modules can act together as a cohesive whole.

Ironically, the DRNNM recognition process, from a high level perspective, does compare well with established or current ideas on biological, visual recognition systems. These systems generally divide the visual system into two distinct areas. The first of these areas (Occipital Level) is the primary visual cortex in the occipital lobe.
which is responsible for low-level processing and the extraction of characteristic features. The second area (Parieto-Temporal Level) processes these features in two parallel streams: the dorsal stream and the ventral stream. The dorsal stream forwards information to the parietal lobe for target location and is commonly referred to as the where pathway. The ventral stream passes information to the temporal lobe for target recognition, the what pathway. Both the posterior and anterior regions of the temporal lobe are involved in storing visual prototypes. It is believed that the posterior region receives input from the parietal stream and is involved in geometric transformations that enable incoming signals to be associated with stored prototypes.

Occipital Level

In figure 3.7 the DRNNM is associated with a biological representation of the visual system. The first occipital level is the “retina”. It responds to visual stimuli and is composed of excitatory neurons that propagate signals back to the cortical maps. The cortical maps consist of minicolumns that are believed to perform a form of principle component analysis on the incoming data. The first of these maps extracts visual orientation features whilst the second extracts invariant angles between features.

In the DRNNM, the “retina” corresponds to Pattern Activity Space. Pattern Activity Space provides initial processing of incoming data with the decay factor used by Pattern Activity Space being equivalent to a tendency for presynaptic cells to continue to discharge after initial firing.

Feature Space is equivalent to the cortical maps found in the occipital lobe. Just like the cortical maps, Feature Space is responsible for extracting features. The cortical
maps are believed to be responsible for features such as; orientation, colour sensitivity, movement and spatial frequency. However, research into cortical map functionality is ongoing and consequently new developments in this area are common.

Parieto-Temporal Level

After processing by the cortical map visual information is processed in two parallel streams. In the DRNNM parallel processing is not performed because the objective of the model is to ensure that recognition is only performed if the data is in a recognizable state. However, although the biological model processes the data in two streams, both the Parietal and Temporal Cortex communicate with each other, when the Parietal Cortex has located the object of interest it notifies the Temporal Cortex, and recognition is attempted. This corresponds to the DRNNM process.

The Parietal Cortex is responsible for determining where important information is in the visual data. It consequently has strong associations with eye movement control and mental rotations. Several researchers have reported that the parietal cortex encodes kinematic variables of visuo-motor transformations [Kalaska, 91][Seal 89][Taira, et al 90]. Other researches have argued that visual mechanisms provided by the Parietal Cortex are the origin of image transformations and have evidence that more time is required to perform larger transformations [Shepard and Cooper 92]. This corresponds to the cyclic transformation processing in the DRNNM. The larger the data transformation that the DRNNM is required to process, the longer it takes to process the data as more cycles are needed to normalize the data. Shepard and Judd (1976) have also found similarities between apparent motion during vision and mental
rotation and Duhamel et al. in 1992 discovered cells in the monkey brain that shift their receptive fields in anticipation of saccades thereby stabilizing their perception to allow for an unstable retinal image.

We can argue that the DRNNM provides similar functionality to the Parietal Cortex through the interaction of Classifier, Decision and Knowledge Space. Classifier Spaces gives an indication of where the object is; Decision Space determines how to focus on that object and Knowledge Space provides mechanisms through which the object can be brought into focus.

The Temporal Cortex associates spatial information with characteristic features so as to recognize what is being presented. In effect, the temporal map associates occipital data with prototypical classes. This functionality is encapsulated by Recognition and Classification Space - recognition prototypes being associated with their corresponding classification. The DRNNM differs slightly from the biological model when handling normalised, or transformed data. In the biological model the storage of transient states is generally believed to be in the temporal cortex. Therefore communication between the Parietal and Temporal Cortex occurs more often than in the DRNNM. The DRNNM does not have a storage issue and can access the transformed data directly from Pattern Activity Space, or directly after a transformation. In a biological system this would not be plausible or possible. The DRNNM could be redesigned to emulate this functionality but this would reduce some of the DRNNM functionality and would provide no obvious benefit.
Figure 3.7 DRNNM Biological Plausibility Model
3.7 Conclusion

In this chapter we discussed the theoretical requirements of a recognition model capable of effective recognition with minimal class examples. The recognition model achieves this objective through the use of seven functional spaces. These spaces interact to create an environment applicable to the recognition data's domain, allowing novel data variations to be accommodated with minimal training examples. Recognition is consequently regarded as a dynamic process that is dependent upon an appreciation of the data's origin.
Chapter 4 - Implementation Aspects of the DRNNM

4.1 Introduction

In this chapter, we discuss the general technical aspects of the Dynamic Recognition Neural Network Model outlined in the previous chapter. This involves the development and implementation of the various modules, and how they interact to enable robust recognition. We discuss the practicality of this module and its recognition potential.

4.2 Dynamic Recognition Neural Network Model Implementation

Overview

A major feature of the DRNN is its data normalisation process. Incoming data is continually adjusted to allow for data transformations, providing a consistent framework from which recognition can be performed.

The Data Normalisation process can be divided into two sub-processes: Holistic Normalisation and Local Normalisation.

- Holistic Normalisation refers to the normalisation of the entire data instance.
Local Normalisation refers to the normalisation of subcomponents, or subsamples, of the data instance.

The Holistic process consists of five different modules that interact to progressively adjust the initial data to within reasonable recognition tolerances. The Local Normalisation phase is a component of the recognition process and is responsible for making allowances for non-isomorphic transformations, such as tearing, or missing data.

The need for both forms of data normalisation is explained in the following example. Consider a grasshopper splattered over a car windscreen. Within the grasshopper remains there will still probably exist some very general grasshopper characteristics, such as a torso, a head and wings. Nevertheless, at best, these remains will only marginally resemble a grasshopper and could be virtually one of any number of insects. The objective of the holistic normalisation component is to enable us to perceive the grasshopper’s remains in something resembling a standard recognition
position. However, it is unlikely that any predefined transformation can adequately reverse the effects of such a traumatic event, consequently we require a more focused positionally relaxed approach to enable the various detached and mangled parts to be tentatively reassociated - a local normalisation system. The local system links together the detached grasshopper features allowing us a greater chance of associating the disparate pieces with a grasshopper.

An alternative approach would be to reverse this holistic to local process by using a series of DRNNs. Different DRNNs could then be used to provide specific, local recognition, with a final DRNN associating the different local DRNN responses, creating a DRNN Hierarchy. In other words, each DRNN would isolate and classify different components and a subsequent higher-level DRNN would then proceed to reassociate the different defined components in acceptable ways. The reassociated components could then be classified using traditional methods.

The implementation of the each of the modules is discussed below
4.3 Pattern Activity Module

Figure 4.3 Relative position of Pattern Activity Module in the DRNNM

4.3.1 Introduction

The Pattern Activity Module is the template upon which all data transformations, feature extraction, classification and recognition are performed. It represents the pattern activity at a particular point in time. Therefore, its structure is dependent upon the type of data being processed and the activity at a specific location.

4.3.2 Development

The modules operation can be summarised as follows:

1. Initially we have a data sample $X$, where $X = \{x_1, x_2, x_3, ..., x_n\}$

2. The initial activity, at a specific point $y$, will be

   $y_n = x_n$

3. Therefore the Pattern Activity for a input sample can be represented by $Y$, where
\[ Y = \{ y_1, y_2, y_3, \ldots, y_n \} \] and \( n \) is the data size of the original data sample \( X \).

4. The Pattern Activity \( Y \) is now processed by the rest of the DRNN. The DRNN may
   - determine that further processing is not required.
   - attempt to recognize the data.
   - adjust the data so as to increase its potential to be recognized by the recognition system.

5. If the DRNN decides that further processing is required and adjusts the data sample then the adjusted data sample will be processed by the Pattern Activity Module. If the Activity Decay is not zero than the data’s previous state(s) will contribute to the Pattern Activity. Therefore after each recognition cycle the pattern activity at point \( y \), will be:

\[
y''_i = \left( T_i(y''_{i-1}) - y''_{i-1} \right) D + T_i(y''_{i-1})
\]

where

\( T_i \) is the selected Knowledge Transformation at time \( t \) (the adjustment made to the previous data) and \( D = \) Percentage Activity Decay per recognition cycle.

Therefore, the activity of any point in time will be equal to the difference between the point’s decaying activity and the new data instance generated by the Knowledge Module’s transformation of the previous pattern activity.
4.3.3 Discussion

The selection of a decay rate is dependent upon the data characteristics. If the data is binary then there will exist definite boundary changes after each transformation. The decay function provides an avenue for minimising the abruptness caused by a data transformation. It reduces the negative affects of the Binding problem by maintaining a link with the previous data state. Alternatively, if the data is of a more continuous nature, then the selection of the effect of the decay function becomes more difficult. It may not be desirable to reduce the values of previous pattern areas by a predefined percentage. However, there is no reason why more complicated schemas cannot be used. These might involve only reducing certain values or particular configurations, e.g. the outline of a shape, or a reduction only in the red component in a colour image.

The effectiveness of the activity decay is dependent upon the extent of the transformation applied by the Knowledge Module. The larger the transformation, the less meaning previous activity levels have as their association with the more recent pattern instances is reduced. For the activity decay function to be effective there should be an overlap between the different data samples, before and after a transformation.

The activity decay is also dependent on the type of features being used. If the features are dependent upon sharp data transitions (high frequency information) then the activity decay effect may not be desirable, as it may inhibit feature interpretation.
4.4 Feature Module

Figure 4. 4 Relative position of The Feature Module in The DRNNM

4.4.1 Introduction

The Feature Module is concerned with the extraction of appropriate classification and recognition features. What these features are depends on the data domain. They may be as simple as raw data values or the product of a basic data normalization process such as a linear transformation, or as complicated as a person’s eye in face recognition, or the relative position, length and angle of a line in object recognition.

The function of features is to allow a more abstract data representation. This reduces the amount of data being processed and usually involves a methodology for extracting useful measurements from the data – a feature extraction process. A simple example is the selection, or a weighting of certain pixel locations in a data array from a character recognition scan. This high-dimensional data vector could be modeled by a variety of techniques, e.g. projecting the selected data points onto a smaller subspace -
MLPs [AT&T], Principal Component Analysis [Turk] and Projection Pursuit techniques [Intrator 92] adopt this approach. Alternatively, we could identify and extract structural features, such as edges [Yuille], providing an equivalent overall data reduction. The problem is deciding upon which and what type of features to use.

A way of minimising the difficulty in selecting suitable features is to ensure that they are independent of the decision making process. In this way it becomes possible to trial different feature sets with minimal disruption to the rest of the system; consequently a selection can be based upon demonstrated results rather than on assumed characteristics. This approach differs from Genetic and Neural Network Algorithms/Methods in that the data is not involved in determining the features. Instead the data domain is used to determine what features are applicable. In this way spurious data does not influence the development of classifier associations. This is not to suggest that features should not be optimised for a data set, it just means that data involvement may reduce feature generality, increasing problem specificity.

Feature segregation also allows different features or feature sets to be used within a recognition domain. Data can then be classified concurrently by different features and feature types without complicating the system design - each feature type being logically and physically separated from each other feature type, thereby minimising potential learning confusion. Feature evaluation can still be accomplished by systematically monitoring the responses from the different feature sets and investigating significant correlations for potential data association or discrimination. This approach has some basis within cognitive science with respect to the “The What and Where” vision theory [Jacobs] [Reeke] and has similarities with sensor fusion.
The recognition process can also benefit by having access to more than one feature set. It is not unusual for some recognition problems to require more than a single feature set to adequately encapsulate all meaningful information. For example in Face Recognition, lighting variations in training, test or recognition data can cause considerable problems. The changes in luminance can significantly affect the feature set causing unexpected misclassifications. The detection of lighting variations, and/or the use of feature sets that compensate for lighting changes may reduce recognition problems. If only one feature set is used the lighting changes themselves may be used as a form of discrimination or association. This situation occurred with the legendary “tank finding project” conducted by the US Army and shown in the BBC’s “The Dream Machine” video - the presence or absence of tanks was based upon the amount of light in each image, which was not the objective of the exercise.

A number of problems may occur with multiple feature sets. These include increases in the computational load, as each feature set has to be extracted and processed and decision making problems can occur when determining which conflicting feature sets’ response to use.
4.4.2 Development

Features are domain specific. In the case studies discussed in chapter 5, the feature sets are non-specific, in order to demonstrate the system's applicability to different domains. In these cases normalised data [0-1] is used for classification, with recognition data consisting of both a specialised local technique and general holistic routines.

The Local Technique used was a simplified version of a matrix lookup method [Crowley 94b]. This method was originally designed to provide rapid retrieval of local invariant features from a facial image. The use of an holistic normalisation routine allowed this approach to be simplified as it was no longer necessary to accommodate holistic variations.
The method consists of a series of Primitive and Complex feature sets. The Primitive features are combined so as to enable indexing into a predefined primitive feature matrix. The complex features generated by the Primitive Matrix are then used to index into a complex feature matrix. This provides a Hierarchical Recognition Schema with successive feature layers that are specialised for a specific recognition domain.

The features are created by using horizontal, vertical and diagonal components extracted from the local data sample at different resolutions, 3, 6, and 9 pixels in diameter. These primitive features are then used to train an elongated Self-Organising Feature Map [Kohonen] – 3 by 400 units. The feature classifications are then used in the formation of a primitive feature matrix.

![Diagram of feature complexity](image)

**Figure 4.6 Increasing feature complexity from combining primitive features**
The use of Primitive and Complex Features allows consideration of a system's relative feature complexity. The more complex a feature, the more specific it becomes to a particular recognition task and the more difficult it is to derive, update and change. For example, if a system consists of large convoluted features, then these features will generally take more time to train, be more specific and less useful for other recognition tasks. Examples of this are the weights connecting a hidden unit to the input layer units of an MLP. These connections form a complex decision region that as the system learns; focus on specific locations within the data samples. These locations create the features that the system will use to discriminate and associate data instances. If you back-project the first layer weights from an MLP into the data domain you can gain a rough indication of the "features" the MLP is using. Unfortunately, the formation of these location specific features may impede further learning as considerable change may be required for the creation of new associations. It is even possible that the initial learning process may prevent the system from ever learning certain features because of the difficulty in escaping a local minima. The old adage "You can't teach an old dog new tricks" comes to mind.

The modules functionality can be summarised as follows:

1. We have a data sample \( Y \), where \( Y = \{y_1, y_2, y_3, ..., y_n\} \), is the processed data from the Pattern Activity Module

2. We have a set of Feature Extraction processes, \( P \), where \( P = \{p_1, p_2, p_3, ..., p_n\} \) that are provided by the Feature Module. Each Feature Extraction process, \( p_i \), represents a procedure for extracting feature information.

3. We apply our Feature Extraction processes, \( P \), to our data sample \( Y \) producing a set of feature vectors, \( F \), where \( F = \{f_1, f_2, f_3, ..., f_n\} \) and \( f_i = p_i(Y) \).
4. The set of feature vectors, $F$, is then forwarded to the Classifier Module for
classification of the data's apparent state.

4.4.3 Discussion

The objective of the Feature Module is to separate the features from the rest of the
system. This permits:

i) the changing of the present features, without affecting the complete system -
although some system compensation may be necessary;

This is possible because the system’s knowledge is independent of the feature
set. The features only provide a possible interpretation of the data, for a
particular context. Features are predominantly used to provide an indication of
the present condition of a pattern, rather than as a means of directly recognising
a pattern.

ii) the introduction of new features to extend the present feature space without
requiring complete system retraining;

In traditional systems, the introduction of new features will require retraining of
the system to allow it to accommodate this new information with respect to
previous learning. This could cause a loss of information, and is generally time
consuming. In this model, learning is a side effect of system development,
which means that new features add to the system’s evolution, providing another
means for the system to interact with the data.
iii) the features to be varied, to optimise classification within different environments.

With difficult problems there are generally a large number of potential features to consider. Consequently the selection of the most suitable features is a difficult task. The features that are suitable for one particular environment or recognition context may be inappropriate in another context. Different classes under different circumstances may be recognised more effectively by different features [Watan]. Unfortunately, there is no real way of evaluating the features until they are actually being used. Therefore by having access to more than one set of features, it becomes possible to cater for greater data variation.

For example, in a 3-layer MLP the primitive features are the input data; the first set of complex features being the values generated by the first layer weights that connect the input layer to the hidden layer. Initially, these complex features are inadequate, but as the system learns, the weights change increasing and decreasing the emphasis placed on the input data locations and their associated value. This causes the complex features to change and ideally improve the classification of the input data. If, however, training is continued with new data, then previous learning may or may not be affected, depending on how the new data affects the weighting of different input data locations. It is possible that a more generic set of features will be found, but it is also possible that previous learning will be lost. It therefore seems reasonable to develop a new feature set for the new data, in that way existing learning will not be affected. The new
feature set may still incorporate the old features; it is just not allowed to change
them, for example the MLP variant Cascade Correlation [Fahlman 90].

iv) data to be processed at different levels by different feature sets.

It becomes possible for data to be classified concurrently by different features
and feature types without complicating the design.

v) the transference of these features to other systems, if appropriate.

Because the features are used to interpret data they can be readily used
elsewhere if applicable. This ensures evolution of the available feature set as the
system learns and develops.

4.5 Classifier Module

![Figure 4.7 Relative position of The Classifier Module in the DRNNM](image)

96
4.5.1 Introduction

The Classifier Module identifies the general state of the recognition data. It consists of a number of different classifiers that try and determine general domain variations. By classifying the data’s state it is possible to make allowances for the effects that state has upon the data, and to improve the system’s recognition capabilities.

In general, the variations that exist in a data set are caused by the accumulative effect of holistic transformations and local topological changes. The number of possible individual transformations is generally small, e.g. most image recognition problems only have to deal with minor rotations, translation and scale problems. However, it is quite common for these transformations to form various combinations, that when combined with local topological changes, make it very difficult to ascertain a data sample’s recognition class when using traditional classification/recognition approaches.

By using a distinct classification module we reduce the complexity of the recognition task by decreasing the possible data variability. Therefore, the recognition system only has to learn to classify good examples of the data class. In its simplest form any data that is not considered to be of “good” quality is ignored by the classification module and consequently not processed by the recognition module. This means that the recognition module does not need to establish associations between different instances of the same class. It only needs to learn how to distinguish between different classes.
From a practical perspective, if a classifier has difficulty learning to classify a data sample’s general state, it is unlikely that the system will be able to determine the actual class of that data sample - the rationale being that the data will either have characteristics of that state, or it will not. If the system is not able to find associations between different classes that define a particular state than how is it going to associate all possible class instances with a specific class and still discriminate that class from other classes?

The classifier’s role is simply to determine whether a particular state is possibly evident in the data. The classifier does not have to be exact or precise; it just has to give an indication. If the indication is wrong, the system has other classifiers that may still provide sufficient information to allow some accommodation to be made. This allows the system to potentially back-track (reverse a previous transformation decision), or to ignore a particular classifier’s response. In the worst case the recognition process will terminate. This ensures that recognition will only be attempted with data samples that can be accurately recognized.

4.5.2 Development:

The development of the Classifier Module involves two phases: a Training Phase and a Classification Phase.

4.5.2.1 Training Phase

1. Given a set of features $F$, where $F = \{f_1, f_2, f_3, ..., f_n\}$, that have being extracted by the Feature Module from the data provided by the Pattern Activity Module.
2. We extract a transformation \( t_i \) from the Knowledge Module for the specific feature type. An alternative operation is to perform the transformation \( t_i \) on the Pattern Activity Data and then proceed through the feature extraction process.

3. We then apply \( t_i \) to \( f_i \). Let \( c_i = t_i(f_i) \). If \( \|f_i - c_i\| > \beta \), where \( \beta \) represents a similarity threshold, then \( c_i \) is a new instance of \( f_i \). The method used to compare the two feature sets can be any applicable metric. In general, the Euclidean distance is sufficient. However different feature types may require different distance metrics.

4. The next step depends on:
   a. If \( c_i \) is a new instance, then \( c_i \) is added to \( C_i \) where \( C_i \) represents the training set that a classifier will use to learn to recognise that transformation for that feature set. How that learning is accomplished depends on the type of classifier being used. The classifier training process may occur concurrently or as a batch process after all the training samples have being processed.
   b. If \( c_i \) is a not a new instance, then we repeat step 3 until a sufficient transformation has occurred, such that \( \|f_i - c_i\| > \beta \) will be true.

5. We then repeat steps 1 - 4 until all transformations \( T \) where \( T = \{ T_1, T_2, T_3, ..., T_n \} \) and \( T_i = \{ t_i, t_i', t_i'', ..., t_i^{(n)} \} \) have been performed.

6. After accumulating the various training state instances we can then use any traditional classification system to associate each training instance with its appropriate transformation type.
4.5.2.2 Classification Phase

The Classification Phase is similar to the Training Phase except for the learning of new state classifications.

1. Given a set of features $F$, where $F = \{f_1, f_2, f_3, ..., f_n\}$, that have being extracted by the Feature Module from the data provided by the Pattern Activity Module.

2. We progressively apply each classifier, $C_i^f$ to each feature vector, $f_i$ where $C_i^f$ represents the classifier used to recognize transformation $i$ for feature type $f$. Therefore, $x_i = C_i^f(f_i)$, where $x_i$ is the classifier response to the feature vector for transformation $i$. Depending on which classification procedure is being used the classifier response will be indicative of the extent to which the feature vector, and hence the input data, is affected by the transformation $i$.

3. We should have at the completion of step 2 the Classifier Module’s response $C$ to $F$, where

$$C = \{\{c_1^f, c_1^{f_1}, c_1^{f_2}, ..., c_1^{f_n}\}, \{c_2^f, c_2^{f_1}, c_2^{f_2}, ..., c_2^{f_n}\}, ..., \{c_n^f, c_n^{f_1}, c_n^{f_2}, ..., c_n^{f_n}\} \}$$

This information can then be used by the Decision Module to determine what to do next.

4.5.3 Remarks:

1. Transformations may be added together a specified number of times $\delta$ - composition.

2. Transformations are generally not mixed, e.g. $t_i^f(t_i^f(f_i))$. This is because the objective is to only recognise a general state. If multiple transformations of
different types are allowed then there is the risk of a combinatorial explosion of classifier vectors. However if this seems necessary it is possibly indicative of:

a. inappropriate features.
b. inappropriate selection of similarity measures.
c. the data transformations being too large.
d. the data sampling being too coarse.

3. New data samples are processed using a classification update rule to ensure an even distribution of classification vectors. When a new vector $Y$ is added each classifier is initially tested to determine whether it responds to $Y$. Initially all classifiers should respond, which is explained below - 4.5.4 Discussion. If a classifier does not respond when it should have, i.e. after a transformation $t_i^f$, then the classifier $C_i^f$ is updated with the new data instance formed by the application of $t_i^f$.

4.5.4 Discussion

Once the classifiers have been determined we can then proceed to implement the operators introduced in chapter 3.

4.5.4.1 Continuous Classifier Space Operator

A Continuous Classifier Space Operator can be implemented by making the adjustment size dependent upon the amount of transformation required - Transformation Variance. Transformation Variance refers to the number of discrete transformations required to change a data sample’s present state to the ideal/normalised state. This is based on the concept that the greater the transformation the greater the region of influence of the classifier. Transformation Variance reduces
the number of vector classifiers required by providing a focusing effect; as the classifier's relative transformation distance from the initial state increases, the classifier's response decreases and its area of influence increases. An alternative way of visualising this is to imagine that the system’s resolution decreases as the transformation size increases. This provides a form of optimisation and reduces vector specificity. Consequently, a high classifier response is indicative of only a minor transformation, whereas a small response suggests either a novel data sample, or a large transformation. Therefore we can say that the classifier effect of a transformation is inversely proportional to its extent.

![Diagram of Classifier Response](image)

**Figure 4.8 Classifier Response in relation to transformation distance**

### 4.5.4.2 Associative Classifier Space Operator

The Associative Classifier Space Operator extends the Continuous Classifier Operator by taking into consideration the response from each classifier’s dynamically opposing classifier, e.g. left versus right. This allows greater classifier generality. If the determination of a state is only dependent upon a single classifier’s response, then with novel data the response may not be sufficiently significant to classify that state.
By taking into consideration the opposing classifier’s response it can then be
determined whether or not there exists something there, or whether the response is
just background interference. For this reason it is important that the classifier
responds strongly when near or in the initial state. If the classifier’s response is
strongest when most transformed, then novel data close to the initial state might not
produce a strong response. This is not desirable as the initial state ideally represents
the approximate state the data will most likely be processed in. Therefore, it is more
probable that data variations will mainly exhibit small transformations. On this basis,
greater sensitivity is needed close to the initial state, rather than further away.

![Figure 4.9 Classifier Selection with respect to Classifier Responses.](image)

The classifier coupling operates by comparing the classifier responses to a classifier
response threshold $t$ and a difference measure $d$. If (classifier response $> t$) and (the
difference between the two classifiers $> d$) then the classifier with the least response
dominates, as that is the deficient characteristic.
4.5.5 Functional Properties

The Classifier Module:

1. allows static data representations to be treated like dynamic data.
2. allows separation of the pattern from its environment. The pattern can then be regarded as an individual entity that manifests itself differently in response to environmental/external influences.

4.6 Decision Module

![Diagram of Decision Module in the DRNNM]

Figure 4.10 Relative position of The Decision Module in the DRNNM

4.6.1 Introduction

The Decision Module is responsible for determining whether to:

1. continue data transformations.
2. terminate the recognition process.
3. attempt data recognition.
It consequently has to be able to select an appropriate course of action based on the information provided by the Classifier Module and any past experience.

4.6.2 Development

The development of the Decision Module is dependent upon developing experience with different classifier information. The response of the different classifiers, individually and as a group, must be taken into consideration and an appropriate transformation, or series of transformations selected.

4.6.2.1 Behaviours

The success or failure of different decisions is represented by the behavioural state $b$ in which each series $s$ of successive decisions places the module.

These states are defined as follows, depending on whether the classifier responses are:

1. **positive**, i.e. each set of classifiers indicates that the data is in a normalised state, then the system is positive stable (excited).

2. **indecisive**, i.e. none of the classifiers is responding above the classifier response threshold $t$, then the system is negative stable (bored).

3. **varying** widely after each decision cycle, then the system is unstable (confused).

4. **cycling** through only a few different responses then the system is oscillating (anxious).
increasingly positive then the system is exploring/searching(curious).

4.6.2.2 Initial Development

The decision module learns to associate different transformations with different classification responses by associating the Classifier responses with their applicable transformation. In situations where no reverse transformation exists the Decision Module does not associate the classification with any transformation. It simply flags that particular transformation as being unavailable.

The initial development of the Decision Module proceeds as follows:

1. an initial data sample is processed by the Classifier Module; all classifiers respond, as the sample is in a normalised state

2. the data is transformed, ideally only one classifier responds, although more might if there is an overlap of classification regions. The Decision Module associates the Classification Module vector with the appropriate reverse transformation, or if using Transformation Variance the dominant classifier transformation.

3. The Classification Vector is compared with existing Decision Module transformation class vectors. The transformation class vectors are then adjusted using the same update rule as used by the Classifier Module.

4. steps 2-3 are repeated until all transformations have been performed.
An alternative approach is to use a conventional expert system. This allows the developer more control over the system's responses. The Decision Module consequently becomes a large decision tree where each classifier response is individually mapped to the appropriate point in the Knowledge or Recognition Modules. For example, if the Classifier $x_i$ responds and other conditions are suitable then the decision module simply activates the Knowledge Module’s $x_i'$ transform.

### 4.6.2.3 Learning Modes

The Module can evolve dynamically through two processes:

1. **Short term learning.** This is based upon past decisions to classifier responses in a recognition cycle, allowing back-tracking if a series of selections is unsuitable and affects recognition. This is evident by negative classifier responses after successive decisions. The system records the decisions made after each classifier response and uses this information to determine whether progress is being made.

2. **Long term learning.** This occurs over several recognition cycles when a series of transformations are continually used after particular classifier responses. This provides a form of optimisation, linking transformation series together with specific decision responses.

### 4.6.2.4 Activation

The Decision Module proceeds through a series of steps in response to recognition data.

1. The Decision Module receives the Classification Vector $V$ from the Classification Module. $V$ is then compared with the existing decision classes to determine a course of action.

2. These actions are:
a. If $V$ activates the normalised state class then the recognition system processes the recognition data.

b. If $V$ activates a transformation class, then it is compared with this particular recognition cycle’s previous decisions.

c. If in the last $\eta$ decisions:
   
   i. the inverse transform for this transformation has been made, then its decision oscillation flag is added to the present decision’s oscillation flag. If the present oscillation flag is greater then a predefined amount then the system is exhibiting oscillating behaviour (see Oscillating Behaviour below).

   ii. there has been no improvement in the system stability, then the system is considered unstable/confused (see Confused Behaviour below).

   iii. there has been consistent improvement in system stability, then the system is in search mode, therefore continue.

d. $V$ is appended to a list of previous vectors for this recognition cycle (if any) with the associated transformation decision.

e. The associated transformation (accessed from the Knowledge Module) is then performed.

3. The next recognition cycle commences.

_Oscillating Behaviour_ represents the presence of a discontinuity in the system’s transformation surface. Therefore, after a transformation the data has not been classified appropriately, causing a discrepancy. This shifts the classifiers out of
balance causing the selection of the opposing classifier. The system can temporarily resolve this by:

1. ignoring that classifier set, however this may prevent effective normalisation
2. increasing the size of the transformations to try and move past that point.
3. back-tracking to before that series of transformations and use secondary, or tertiary selected transformation classes to move around the problem area.

In the long term the responsible classifiers need to be updated.

*Unstable/Confused Behaviour* is caused by classifier discontinuities as with oscillating behaviour. The data is classifiable by some of the classifiers and not by others. This causes seemingly confused behaviour as the system intermittingly juggles the data into various positions. The only effective solution is to update the problem classifiers.

*Negative Stable Behaviour* indicates a widespread problem with the classifiers’ classification of the data sample. The data is simply beyond the system’s capabilities. This is possibly caused by inappropriate training, but may be indicative of a novel data sample that possesses new characteristics. It may be possible to retrain the system to more effectively handle this data, but consideration should be given to extending the present feature and classifier sets.

### 4.6.3 Discussion

A problem with this module is determining which transformation to use. Although the knowledge may be available, knowing when to apply it can be difficult. There might be a number of transformations, or rules, that are applicable at any time.
The effect of certain transformations may mask other more critical transformations. This is not necessarily a classifier problem, as the classifiers respond to the dominant classification. It is the decision module's role to ensure that less dominant transformations are appropriately handled. However, it does indicate a dependency on the classifier responses.

### 4.7 Knowledge Module

![Diagram of the DRNNM showing the knowledge module](image)

**Figure 4.11 Relative position of The Knowledge Module in the DRNNM**
4.7.1 Introduction

The Knowledge Module is representative of a domain’s dynamics. It contains all the information required to provide the system with the ability to simulate the data domain’s environment (a priori knowledge).

![Diagram of Knowledge Module]

**Figure 4.12 Pattern Transformation Knowledge**

4.7.2 Development

The type of transformations that are available is dependent upon the data domain. For example, they might consist of affine transformations in an image recognition problem. However, the transformations are not restricted to linear transformations, and may include any form of data mapping. For example, for a sequence of numbers, or series of letters, the transformations could consist of morphing routines that gradually adjust each letter or number. In general, the transformations should be isomorphic to allow transformations to be reversed when appropriate.

4.7.3 Discussion

The main benefit of the Knowledge Module is that it:
1. separates the pattern dynamics from the recognition process - permitting the system to concentrate on recognising the pattern, rather than on trying to:
   a. find associations between different data representations, of what is intrinsically the same pattern
   b. weight different data locations to improve the discriminatory capabilities of the extracted features

For example, “knowledge space” might consist of nothing more than a series of affine transformations in an object vision recognition problem. Although, it is possible that lighting, background, noise and other transformation types might also be present, as these represent different possible factors that could affect the object's representation, and be responsible for different apparent pattern instances.

2. provides the system with knowledge regarding the data domain’s dynamics. This is accomplished by giving access to rules and data transformations that characterise and mimic the data environment.

3. allows associations to be made between data variations and their abstract general classification/categorisation. Data undergoes transformations or changes for a reason. For example, an object might rotate in response to some external or internal influence. Although the object has not changed, the observed representation of this object may be significantly different from the ideal object used in the development of the original vision system. The Knowledge Module provides a means of simulating the possible pattern changes and thereby provides a means of compensating for the effects of these transformations.
4. provides a link between the traditional symbolic world of AI and the sub-symbolic world of Neural Networks. The transformations that Knowledge Space consists of are equivalent to rules. They provide a means of transforming data from one condition to another. They can therefore allow the system to be used in a wider variety of applications other than strictly pattern recognition.

5. allows a less complicated learning environment. As only a single data example is required there is less chance of confusion during training, or complications arising from difficult associations. This means less relative time is required to train a system.

4.7.4 Problems

The inclusion of a knowledge base may not seem applicable in a learning environment as no real learning is actually accomplished. However, this is not strictly true as complex transformations can be developed to reduce unnecessary steps; if a series of transformations are continually being used then these transformations can be grouped together and accessed directly. This dynamic extension of the module involves primitive transformations being used to create more complex transformations.

Other problems that potentially exist with the implementation of the Knowledge Module include:

- The module’s dependency upon having some primitive understanding (transformations) of the environment, or data dynamics. This initial knowledge is
necessary to allow associations too made between transformations and the effect these transformations have on the data. It may be difficult to determine the data dynamics or the applicable transformations in a system. If this knowledge is not provided then the system cannot establish transformation relationships and perform appropriate data transformations. This introduces a system bias towards solutions that are achievable with the existing knowledge. If the knowledge is unavailable then the system’s performance may be less than desirable.

- How to apply these transformations to problems without degrading or further complicating the data? Application of the transformations can distort the data by introducing unwanted artifacts which may affect the recognition process.

### 4.8 Recognition Module

![Figure 4.12 Relative position of the Recognition Module in the DRNNM](image_url)
4.8.1 Introduction

The Recognition Module is responsible for classifying the pattern data.

4.8.2 Development

The objective of the initial modules is to normalise recognition data so as to provide the recognition module with data that is ideally structured and positioned for recognition. The recognition module exploits this to minimise the complexity of the recognition approaches. These approaches can be divided into two main divisions:

1. **Holistic**

   This is the simplest approach. It ignores local variations and uses the entire data sample. Recognition is consequently reduced to simply comparing the data sample with known recognition classes. If the distance between a recognition class and the sample data is less than a defined similarity threshold then the data is classified as belonging to that class. If more than one class is selected then perhaps the similarity threshold is too high, or the class representative vectors are not properly positioned. Nevertheless, this situation can be partially rectified by selecting the closest class, or the class with the most representatives selected - assuming similar class numbers.

2. **Local**

   Under some conditions, data normalisation cannot provide adequate data transformations to make allowances for local topological variations without sacrificing data generality. Therefore, it may be necessary to use a local
recognition approach to compensate for the local topographical variations. This will allow constrained system tolerance to local changes.

There are two types of local recognition systems. They both function by classifying the most recognisable region within a constrained area. Both approaches basically compare all the possible local data sets within a designated search area with a local feature database and select the most suitable classification. As the data is already normalised it is not necessary to establish relationships between local features as their interrelationship has already been established by virtue of the normalisation process. This significantly reduces recognition complexity as it is not necessary to compare a local feature with the entire feature set, but only a small subset applicable to its relative area.

The normalisation process also removes the need to verify the validity of local feature combinations as would be the case with an unconstrained data sample. It is conceivable for an unrecognisable, unconstrained, jumbled combination of local features to provide sufficient local feature correspondences to be seemingly recognisable. This is generally not desirable.

To allow robust local normalisation, the recognition module can be given access to the Classifier Module information via the Decision Module and the ability to perform localised transformations via the Knowledge Module.

Although the classifier information is generally not required, it can be important when there does not exist a suitable transformation to normalise the data. For
example, 3D rotations of facial data can be difficult to accommodate. Therefore, the recognition module can use the classifier information to increase the region search dynamics, through the use of transformations from the Knowledge Module and by adjusting its search parameters. This might involve narrowing the search region and repositioning the initial search areas so as to compensate for the apparent rotation. Specific localised 3D transformations could also be provided that compensate for the effects of the 3D rotations on the features (if the 3D transformations are available why not use them holistically? They potentially introduce artifacts that complicate the recognition process). Another example, where classifier information would be useful is adjusting the feature search regions to compensate for the effect of data artifacts, such as glasses on a person’s face.

The first local approach consists of:

a. The data sample being coarsely sub-sampled in a fashion applicable to the data domain. For example, in Face Recognition a grid centered on the face can be used to sub-sample the data.

b. Each node on the grid represents the initial starting point from which local features are extracted

c. The node area is searched by systematically varying the feature area extracted and comparing this feature area with the local feature database. As with the holistic recognition approach the best match is selected and used to classify that region

d. step c. is repeated until all nodes on the grid have been processed

e. The vector of local classifications is then compared with each of the recognition classes that have been processed in a similar way.
The second approach is virtually the same, except:

a. that rather than the most responsive data from each node being classified it is added to a global recognition vector.

b. after all of the local areas have been added to the global vector it is then compared holistically with each of the recognition classes, the latter having been obtained previously by storing the local feature regions from initial training sampling.

Figure 4.13 Examples of local feature classification.

Note the relative similarity of the points selected even though the faces are in different states.
4.8.3 Discussion

1. Rather than systematically classifying each node, the class data is processed to determine the nodes that provide the maximum discriminatory value. During recognition, these nodes are compared first to determine whether a positive identification can be made based upon them, rather than from using all the grid points. This has some basis in vision research for the rapid identification of visual information [Yarbus][Stark].

2. The search around a grid point is not necessarily restricted to normal domain space, but may also be conducted in transformation space. In the example earlier the grid points were positioned on a person’s face, with the implied search pattern being around the initial point in a 2D fashion. This is not necessarily the case, as the system has access to a variety of domain related transformations in the Knowledge Module. It is therefore possible to search through transformational space by using these transformations to transform the local extracted feature. This allows potentially greater system generality, but also increases the computational load of the system.

3. A third possible recognition approach is for the recognition module to become a localised series of DRNNMs. Rather than simply classifying each local feature created by our sub-sampling, we use a local DRNN to classify local features and an holistic DRNN to associate the different local classifications.
4.9 Classification Module

![Diagram of DRNNM modules](image)

Figure 4.13 Relative position of the Classification Module in the DRNNM

4.9.1 Discussion

The Classification Module is responsible for associating the data with a suitable label. The module enables multiple recognition classes to be associated with a single label.

4.10 Conclusion

This chapter provided a general implementation overview of the DRNNM. This was achieved by discussing the various attributes of each model component and how they can be trained and developed. In the next chapter results obtained from using the DRNNM are discussed.
Chapter 5 - Model Implementation Results

5.1 Introduction - Image Recognition

In this section the DRNNM is applied to two difficult image recognition problems: handwritten digit recognition and human face recognition. These problems were selected as they are both good examples of the type of problem this thesis is addressing. Where possible the DRNNM results are compared with the reported findings, and the results from other applicable models.

5.2 Digit Recognition

The recognition of handwritten digits is difficult because of the amount of variability that exists within each digit class. This makes associating different class instances troublesome as the correlation between digit instances may be extremely low; necessitating that the recognition system be capable of developing non-linear class instance associations. This situation may be further complicated by a lack of sufficient examples to fully characterise the digit parameter space and to enable the initial development of suitable class associations.

Ideally, a digit model should be developed that characterises the data variability; however this is not computationally feasible, consequently digit/character recognition systems generally try to use structural feature information. This reduces the dimensionality of the problem to more manageable levels. For example a system
might use the relative line position and intersection information. However, digits are not restricted to clearly defined lines and may be distorted in a number of ways from an "ideal" representation. This presents a significant problem in that it may not be possible to find the desired structural features to classify the data.

The approach adopted by the DRNNM is to exploit the digit's 2D data representation. This means that we can categorise most of the distortions that affect digit, or image, recognition by combinations of 2D affine transformations. The problems that occur from digitization, such as the addition of extraneous pixels, will generally not affect most systems, unless the digitization effects are excessive, e.g. greater than 15% of the data instance size - even crude systems can handle Signal to Noise Ratios of 10%. It is the accumulative effect of the many possible transformations that generates the large number of possible variations. For example, a digit might be slightly rotated and translated, with a marginal warp and/or a change in scale. These transformations make it difficult to associate different digit instances with the desired class and to fully characterise the digit possibility space. However, the Knowledge Module of the DRNNM can generate these types of transformations and thereby enable the DRNNM to search for the appropriate class by selectively applying suitable transformations. This simplifies the DRNNM's training as consideration does not have to be given to all the possible digit variations. The only real difficulty is classifying the digit's state to optimise the search process.

To enable the classifiers to efficiently determine a digit's state, the effects that different transformations have on the different digits have to be generalised. It is possible to train the classifiers by systematically applying 2D affine transformations
to each digit representative(s) and then training each classifier to recognise the characteristics of each state. This process can be difficult as the digits are all intrinsically different and therefore the classifier associations with any particular state may become highly localised. The DRNNM avoids this complexity through the use of Transformation Variance, classifying the relative state rather then the absolute state.

5.2.1 Module Initialisation

1. Pattern Activity Module

   The Pattern Activity surface was a square, 16*16 units in size, with an activity decay factor of 100%.

2. Feature Module

   The features used were real values between 0-1.

3. Knowledge Module

   The knowledge provided consisted of 2D affine transformations.

4. Classifier Module

   The states the Classifier Module was trained to be responsive to where the same 2D affine transformations that the Knowledge Module was given access to.

Classifier training consisted of presenting the different classifier sub-modules with examples of their applicable digit state, the ideal state and the opposing state. This provided a domain specific state contrast mechanism through which to normalise the digit. For example, if the classifier was being trained to recognise a
left translated digit, then training would consist of positively reinforcing left and ideal state digit instances, whilst negatively reinforcing right related states. This creates an overlapping region between different classifiers that enables the creation of a transformation space. Without this overlap classifiers can become isolated, inhibiting smooth transitions from different states during the normalisation process.

5. **Decision Module**

The Decision Module was used to provide a checking function and to select an appropriate transformation. It determines system progress by checking the time (number of cycles) used to recognise the digit, the classifiers’ responsiveness and the system’s perceived confidence in the data being a digit. If none of the classifiers responses are above 40% we can assume that we are not dealing with a recognisable digit. The selection of a transformation was accomplished by associating classifier vectors with different transformations.

6. **Recognition Module**

The Recognition Module classified the digits by using a local-holistic recognition process that compares the transformed digit with the initial training digits (Figure 5.1) The local-holistic recognition system corresponds to the second local recognition option discussed in Section 4.8.2. The first local recognition approach was found to be unsatisfactory with digit data due to the artifacts introduced by the normalisation process. The local features used were 5x5 regions spread over an evenly spaced 6x6 grid on the images surface. Within each graph node the
local feature with the greatest activity was selected and added to the recognition vector - the activity measure being the sum of the variations from the norm for the feature area. A number of other approaches exist [Malsburg 88] [Rybak 92b].

7. Classifier Label Module

The Label Module simply contains the class identifiers {0, 1, 2, 3, 4, 5, 6, 7, 8, and 9}.

5.2.2 Training

1. Training Examples

These consisted of a series of handwritten numerals from 0-9.

![Figure 5.1 Training Examples](image)

2. Test Examples

Examples of some of the digits tested are shown below.

![Figure 5.2 Testing Examples](image)

Training consisted of each classifier learning to recognise a different type of digit transformation. In all 17 modules were trained, with each module being responsible for one of the following characteristics: a translation to the: left, right, up, down, up-
left, up-right, down-left, or down-right; or, a rotation: to the left or right; or, a scale change: of greater, or less than normal.

For example, the following characters illustrate some of the different states the system was trained for. A variety of characters in different states and state extents (e.g. small, medium or large: translations, rotation, scale variations.) were used to enable the system to generalise character transformation trends. The objective was for the system to learn to recognise different dominant states and not to be concerned with the recognition of the character.

![Figure 5.3 Data Variation Type Examples.](image)

NB. Although the digits in Figure 5.3 are different representations of the same digit and look relatively similar from an automated recognition perspective, they are very different. If we compare the various digits with the centered digit, the recognition confidence is very low.

The digits were dithered down from an initial size of 32x32 pixels to 16x16. This minimised abrupt state transitions and reduced the extent of state transitions caused by small variations, providing a smoother transformation surface.
To recognise the different digit states, MLPs consisting of an input layer of 16*16, a hidden layer of 5 Units and an output layer of 1 were initially used. The MLPs were used because they provided an effective means of minimising the effect of noise and spurious data and allowed the development of non-linear associations between the different state training instances. This was considered particularly important as each digit is intrinsically different - complicating the classification of general states.

The training of each MLP consisted of the network learning to distinguish between its determined state and opposing state. This approach was found to be easier than trying to teach the network to recognise each particular instance of a character, or trying to extract consistent structural features. An advantage of this approach was that it was relatively easy to automate, with the transformation of each digit being linked to an appropriate classifier - as discussed in Section 4.3.

Although this training approach reduced learning complexity, each Classifier's learning task is still difficult. The network still had to be able to find sufficient correlations between the various digit states to learn a particular data trend, or to associate the data non-linearly. This necessitates the creation of complex decision regions, which is not really desirable in this instance. Remember the objective of each Classifier module is only to give an indication of a particular state; if too much emphasis is placed on specific learning the module's effectiveness is reduced, as its learning becomes more tightly bound to certain locations. The dithering of the digits was found to be useful in blurring the sharp distinctions between digit states and reducing this dependency. The possible loss of transformation accuracy caused by
dithering was compensated for by the overlap of the classifier modules learning regions and the interaction/contrast with the opposing state.

After each Classifier Module was trained their responses were linked to the applicable regions in the Decision Module. In general, the Decision Module would select transformations on the basis of the most responsive classifier.

This initial system appeared to perform well normalising 80% of the digits tested. However, complications became evident with the normalisation of “0” like digits. Attempts to rectify this situation with a greater number of hidden units, 10 - 20, tended to cause the system’s learning to become specific causing greater sensitivity to transformations. To compensate for this situation a further set of classifiers was introduced that took into consideration the activity within different sections of the image plane - exploiting the binary nature of the digits. Basically the image region was divided up into nine equal sized areas with different activity combinations within each section being used to define different possible states. This approach was found to nicely complement the existing MLP classifiers improving general digit normalisation.

5.2.3 Remarks

At the end of each digit set the DRNNM recognition results have been contrasted with a standard holistic character recognition system and an MLP system based on
LeCun’s design[LeCun]. LeCun’s system is noted for its recognition of US ZIP codes.

Both contrasting systems were trained using the specified training set with learning variations being created by systematically translating, rotating and scaling the training digits, to varying degrees and in different combinations.

The holistic system was trained using LVQ [Kohonen] with each training digit representing a class vector. To provide a realistic appraisal of this system’s recognition success, the confidence measure was calculated by contrasting the average class activity with each recognition class response. Consequently high system confidence is only evident when there is a substantial difference between the recognised class response and the other class responses.

The MLP-Based system consisted of a standard MLP with 256 input, 30 hidden and 10 output units. As anticipated in section 2 - Problems with Learning Complexity - this system had difficulty learning suitable mappings under the training conditions; after a number of restarts and a considerable period of training time (days) the system still had not converged. Consequently, to reduce learning complexity a variation of this system was used. This involved normalising the data with the DRNNM normalisation process before commencing recognition.

5.2.4 Results - Digit Recognition

Where applicable, some normalisation steps have been summarised into a single step.
Single Digit translations (1/16 units) only are made, with rotations being set at 15 degrees and scale changes restricted to 10 percent for a classification cycle. The activity decay factor is 1.0

5.2.4.1 Digit 1

This set of results demonstrates the transformation of a class instance, the digit “1”, from an initially unrecognisable state, to a more recognisable state over three recognition cycles. It is possible to observe considerable improvement in recognition confidence after the second cycle.

![Figure 5.4 Normalisation of Digit 1](image)

Figure 5.4 Normalisation of Digit 1
Digit 1-1 **decision**: rotate the digit to the left

Digit 1-2 **decision**: rotate digit to the left.

Digit 1-3 **decision**: reduce digit’s size

Digit 1-4 **Decision**: attempt recognition (all responses shown to indicate system’s positive response)

**Figure 5.5 Classifier Responses for Transformation of Digit 1**
Recognition (Digit 1)

Figure 5.6 Holistic Digit Recognition Results for Digit 1

Figure 5.7 MLP-Based Digit Recognition Results for Digit 1
5.2.4.2 Digit 2

In this example the digit 2 is normalised over a series of recognition cycles. Although it never achieves what could be considered an ideal representation of this digit it nevertheless becomes gradually more recognisable to the system.

Figure 5.9 Normalisation of Digit 2
Digit 2-1 **decision:** Translate digit up and to the right.

Digit 2-2 **decision:** The selection of a possible translation is confusing ((down right, left) = down + up = cancel), therefore select the next predominant choice and rotate digit to the left.

Digit 2-3 **decision:** Translate digit left.

Digit 2-4 **decision:** Scale digit down

Digit 2-5 **decision:** Translate digit down and to the left.

Digit 2-6 **decision:** Translate digit down and to the right.
Digit 2-7 **decision**: As a translation, up and a translation left, (interesting that upleft classifier is responsive) is simply a reverse of the previous step, down right, the system decides to scale the digit down.

Digit 2-8 **decision**: Rotate and shift the digit to the left.

Digit 2-9 **decision**: rotate digit to the left

Digit 2-10 **decision**: Rotate right is the reverse of the previous step, therefore ignore and try a translation up, and to the right

Digit 2-11 **decision**: rotate digit to the left

Digit 2-12 **decision**: No further significant responses

**Figure 5.10 Classifier Responses for Transformation of Digit 2**
Figure 5.11 Holistic Digit Recognition Results for Digit 2 – no significant recognition result.

Figure 5.12 MLP-Based Digit Recognition Results for Digit 2
5.2.5 Miscellaneous Digit Normalisation Results

a) 

b) 

selected = 3
The digit normalisation examples in Figure 5.14 illustrate the normalisation of a series of handwritten digits. More detailed information regarding these digits can be found in Appendix A.

5.2.6 Comments.

In the classifier responses shown we can observe on occasion rapid fluctuations in some of the classifier responses following a data transformation. For example, if we look at the Digit 2 classifier result set, 2-7 and 2-8, an abrupt reversal is evident in “up” and “down left”, classifier pairs.

This is indicative of a number of different potential problems:
1. The knowledge module’s transformations are too large

2. The classifier learning is too specific and consequently localised activity regions have formed.

3. The resolution of the digits is too low and consequently digit transformations introduce very abrupt state changes.

These abrupt classifier transitions may constitute a problem if they cause the system to become unstable. This is because abrupt transitions make it difficult to determine subsequent transformations due to, too many significant classifier responses. Ideally, a simple transformation should not result in a number of classifiers suddenly undergoing considerable change. This introduces the very real possibility of complex classifier cycles forming that could inhibit the system’s effectiveness.

On a more positive note the results demonstrate the advantages of using a modular design. The failure of any of the system’s modules is clearly visible and consequently appropriate measures can be taken.
5.2.7 DRNNM Digit Recognition Summary

Figure 5.15 shows the “general digit” confidence of the system over an averaged series of recognition cycles. There is a general increase in confidence after each recognition cycle indicating that the DRNNM is gradually improving the “quality” of the digit being recognised.

The results indicate that the DRNNM can improve the recognition of handwritten digits with the use of only minimal training examples. In these tests no attempt was made to exploit the digits’ characteristics with the test data bearing little resemblance to the original training digits.
With the use of only one class instance the DRNNM did demonstrate difficulty with recognising hollow patterns, such as “8”, see Appendix A. Interestingly though the normalisation of these digits was still reasonable. Recall that the objective of this test case was to demonstrate the ability of the DRNNM to robustly recognise digits with minimal examples, not to provide perfect digit recognition. On this basis, the DRNNM has performed well, normalising to some extent all of the digits tested. The recognition of these digits, as shown in the examples, was positive, with the contrasting systems failing to respond as confidently/significantly in most cases.

Overall the digit recognition was 80% with the rest being disregarded as unrecognisable. This is a relatively low figure when compared with other systems. However, when more class instances were used for recognition this result rapidly became more competitive with recognition results of 95% - only a few examples not being classified correctly.

The recognition results in themselves cannot be considered as a good measure of the system’s success. This is because there are only a few digit classes and they are all distinctly different. Consequently, after minor normalisation it becomes relatively easy to distinguish the different digits, especially when mathematical morphology is used to reduce a digit to a constant skeletal form. The normalisation results however are very important, especially when applied to an area such as Face Recognition where there is virtually an infinite number of very similar classes.
5.3 Face Recognition

The problems inherent in Face Recognition are different from those encountered in digit recognition. These include:

1. a very large set of classes
2. an environment where each class is potentially similar to another class.

From a recognition perspective this means that we are trying to recognise a virtually limitless set of similar patterns. This is computationally not necessarily a problem, if it is possible to clearly separate each class. However facial classes are extremely variable, which means that they do not form tight class clusters, but are spread throughout large areas of the total facial class possibility space. Interestingly, the majority of systems do not try to cater for this variability; they generally rely on the high dimensional nature of the data to provide suitable class separation [Turk] [Kirby 87] [Midorikawa] [Makoto]. This represents a serious recognition problem as it is possible that different facial states may result in misclassification with certain facial types. Unfortunately, this is largely dependent on the face and the type of change, or changes, that are presently affecting the face’s presentation. However, as the face is unknown, it is difficult to determine the state the face may be in and consequently suitable accommodations cannot be made. This reduces the recognition process to a dependency upon class similarity measures which are fundamentally flawed, in that for the system to generalise sufficiently well to cater for probable class variations it must also become vulnerable to misclassifications.
5.3.1 Module Initialisation

A face database was created to enable testing of the facial recognition system. This database consisted of 70 different people’s faces with 10 different perspectives of each person’s face – a total of 700 facial images. These facial images consisted of grayscale images, 64x64 pixels in size.

The grayscale images were rescaled to range between 0 and 1 by using a standard linear transformation. Each input variable being treated independently, with each variable \( x_i \) having its mean \( \bar{x}_i \) and variance \( \sigma_i^2 \) calculated with respect to the training set, e.g.

\[
\bar{x}_i = \frac{1}{N} \sum_{n=1}^{N} x_i^n
\]

\[
\sigma_i^2 = \frac{1}{N-1} \sum_{n=1}^{N} (x_i^n - \bar{x}_i)^2
\]

where \( n = 1, ..., N \) are the faces used in training. This allows us to define a set of rescaled variables by using

\[
\tilde{x}_i^n = \frac{x_i^n - \bar{x}_i}{\sigma_i}
\]

The rescaling training set consisted of 10 randomly selected frontal faces.

1. Pattern Activity Module

The Pattern Activity surface was a square, 64x64 units in size, with an activity decay factor of 100%. The use of different decay factors did not produce any significant difference in the classification process.
2. **Feature Module**

The features used were real values between 0-1.

3. **Knowledge Module**

Like digits, faces are affected by 2D affine transformations; therefore the Knowledge Module contained the same types of transformations as were used in the digit recognition example.

4. **Classifier Module**

The Classifier Module was trained in a similar fashion to the digit recognition Classifier example. A few new classifiers were introduced to allow three dimensional face states to be classified. These consisted of: “looking up”, “looking down”, “looking left” and “looking right”.

The training consisted of randomly selecting 5 frontal face views from the 70 possible frontal faces. These 5 faces where not used for testing of the classifiers or in the recognition component of the system. The three dimensional perspectives of these five faces were also used to train the system to recognise three dimensional states.
Figure 5.16 Frontal faces used in training DRNN system

5. **Decision Module**

The Decision Module was used to determine which transformation to use and to enable backtracking to an earlier system state after a series of transformations proved ineffective. This was virtually the same as that used for in the digit recognition example.

6. **Recognition Module**

The recognition approach used was the same as that used in the digit recognition example, the second holistic - local method outlined in Section 4.8.2., the difference being in the local features used. These were 7x7 regions spread over an evenly spaced 5x7 grid on the central 42x56 unit portion of the original 64x64 image surface. Figure 4.13 shows examples of the grid deformation for different facial variations. The creation of the recognition set therefore consisted of extracting the local features and adding them to a global vector – one global vector for each of the 65 remaining front facial views, thereby generating 65 recognition classes.

The recognition process involved a three step procedure:

a. A “recognisable” DRNN normalised face has its local features extracted, as mentioned above, and added to a recognition data vector.

b. The recognition data vector is systematically compared to each recognition set vector using Root Mean Square. The result for each comparison is saved,
with the recognition set vector that generates the lowest result, or shows the smallest difference, being considered the “winner” and the recognition data vector associated with that recognition set vector. If the “winning” score is above a 20% error then the face is considered not recognised. The error was calculated using \( e = \frac{\sum_{i=1}^{n}|x_i - y_i|}{n} \). Other error calculation schemes could be used, such as classical standard deviation error methods, however this simple scheme was found to be just as effective.

c. The confidence of the system is then calculated by averaging the saved responses from all the vectors and comparing this with the result from the “winning” vector and the next four lowest results.

1. **Classifier Label Module**

The Classifier Label Module contains the identifiers for the 65 faces used and associates the recognition set vector with a specific identifier. The Classifier Module is of greater use when there are multiple recognition set vectors for a class.
5.3.2 Face Recognition Results

This section describes the results from the DRNNM normalisation cycles, with a comparison of the DRNNM response with an Holistic Model, based on Pentland and Turks’ research [Turk], and a Local Model based on Charles Von Malsburg efforts [Malsburg 92].

5.3.2.1 Face Recognition No 1

Figure 5.17 Facial Normalisation

![Facial Normalisation][1]

Face 1-1 Decision - Shift Up-Right

Face 1-2 Decision - Shift Right
Face 1-3 Analysis of Facial 3D perspective is done last.

Figure 5.18 Facial Normalisation Results (No 1)

Figure 5.19 Holistic Face Recognition

Figure 5.20 Local Face Recognition
5.3.2.2 Face Recognition No 2

Figure 5.20 Local Face Recognition

Figure 5.21 DRNNM Face Recognition

Figure 5.22 Facial Normalisation.

Face 2-1 Decision - Shift right - scale down.

Face 2-2 Decision - High Confidence Score
Face 2-4 3-D Classifier Responses

Figure 5.23 Face Normalisation Results (No 2)

Figure 5.24 Holistic Face Recognition
This is an interesting result in that it appears that the local system has performed better than the DRNNM. However, this is not the case - the local system has simply identified the face and has not taken into consideration the face’s 3D state. The DRNNM model is not as confident because the face is not in an ideal state. Although
in this case this has not presented a problem, in the next two examples the effects of disregarding the facial state becomes more evident.

5.3.2.3 Face Recognition No 3

![Figure 5.27 Facial Normalisation](image)

**Face 3-1 Decision - right**

**Face 3-2 Decision - Rotate Right**

**Face 3-1 Decision - Scale Down**

**Face 3-2 Decision - Shift Right**
Face 3-5 Decision - Attempt Recognition

Figure 5.28 Face Normalisation Results (No 3)

Figure 5.29 Holistic Face Recognition
This example illustrates an interesting characteristic of the facial normalisation process that is distinct from digit/character normalisation. With digit recognition the normalisation of the digit virtually guarantees robust recognition. This is because digits are all by definition different. Faces however are all very similar.
Consequently, the normalisation process also potentially improves other recognition class responses. This can be a problem if the facial image is affected by the various data transformations used during normalisation. Some transformations may increase the similarity of some of the recognition classes by introducing artifacts that the recognition system may inadvertedly employ to distinguish the various classes. Alternatively, these artifacts may reduce the similarity of the normalised face to the appropriate class.

The example also illustrates just how difficult it can be to distinguish different facial classes and why there is a definite need for an awareness of the facial state.

5.3.2.4 Face Recognition No 4

![Facial Normalisation](image)

Figure 5.32 Facial Normalisation

Face 4-1 Decision - Shift Right

Face 4-2 Decision - Shift Down Right
Figure 5.33 Face Normalisation Results (No 4)

Figure 5.34 Holistic Face Recognition
Figure 5.35 Local Face Recognition

Figure 5.36 DRNNM Face Recognition
5.3.3 Miscellaneous Face Normalisation Results

The above examples are a series of randomly selected faces that were accurately classified and/or normalised. Figure sets a) and b) illustrate how much a face may
vary even after the normalisation process. An unfortunate aspect of normalisation is that for it to be effective it must be tolerant and consequently there may still be some difference between normalised faces.

5.3.4 Face Normalisation Result Summary

The results given above are only a representative sample. With the test database of 70 people, the system successfully classified those faces considered recognisable. This was 80% of the 700 faces. The remaining faces were generally unsuitable for recognition due to lighting differences, or large 3D transformations.

In general, the DRNNM results were 35% higher than those produced with the comparative local or holistic approach. However, when dealing with good quality frontal faces the local, global and DRNNM recognition responses were similar. Nevertheless, the global approach did tend to generate more (10%) false positives as the quality of the faces being recognised deteriorated.

The DRNNM did fail to normalise faces where there existed a significant difference in the colour, or lighting, relative to the training faces. This represents a significant problem in that lighting variations can be difficult to accommodate due to a lack of regularity. A possible solution to this problem is to use features that are not dependent upon the lighting conditions. This may be possible using edge features, although the sensitivity of Isodensity techniques to light [Nakamura 91] does highlight the difficulty of this problem. Ironically, edge data in itself may introduce other problems due to its need for a high resolution to allow effective facial
discrimination; low resolution high frequency/edge facial images can be very difficult to recognise.

5.4 Conclusion

An important aspect of the results was the reuse of the model structure for both digit and face recognition. Virtually exactly the same system was used for both recognition problems.

The negative effects of the binding problem are visible in most examples. This was particularly evident with the traditional digit recognition approach. Under the testing conditions this system performed poorly, which is indicative of its rigid learning. More recent character recognition approaches circumvent this rigidity by using different models for different handwritten character sets [Li]; the character set being selected on the basis of a country’s style of writing. Therefore, rather than trying to find consistencies across a character, a totally different module is used based upon a model of the dominant writing style. Although this is not aesthetically nice, this is a realistic approach, consistent with these results when using a traditional classification system and is supportive of the multiple feature discussion in Chapter 3.

An important aspect of these results was the generality of the training data. Throughout the tests no attempt was made to use data specific to either recognition problem, or to simplify the learning task. Both systems were trained using simple data to demonstrate the model’s applicability to other recognition areas.
Overall, the DRNNM results demonstrate that the model is capable of recognising dynamic data from minimal static examples. Several examples of digit and face normalisation and recognition were presented and compared with alternative approaches, and found to perform significantly better.
Chapter 6 - Discussion and Conclusions

6.1 Introduction

In Chapter 2 the inadequacies of existing recognition systems was discussed. These problems were related to the inability of these models to robustly recognise dynamic data from minimal static representations. This was perceived to be caused by the binding of the pattern data to the classification structure, limiting system learning to developing associations dependent upon the data values at different data locations. It was argued that this coupling prevented access to, or appreciation of the data’s environment, the pattern data having become an intrinsic part of the actual classification system. We therefore concluded that the essential problem with traditional recognition systems was the lack of an abstraction layer to separate the pattern data from the classifier mechanics. We then determined that the source of the majority of recognition problems, such as learning complexity, learning interference/confusion and classification difficulty with minimal data samples, are in fact different manifestations of the same initial binding problem.

Motivated by this conclusion, in Chapter 3 we began to develop a model that could separate the pattern data from its environment, and consequently from the classification system. We determined that to accomplish this task requires the interaction of seven different spatial operators. These operators enable the recognition system to recreate/simulate the data’s domain and thereby reduce the environment’s
effect on the data’s representation. It was argued that for this functionality to be achieved the recognition system needs to be dynamic and capable of perceiving and manipulating the data. This removes the dependency of the recognition system on discriminating and associating classes on the basis of different data regions. The system consequently has the freedom to explore and actively search for an appropriate class, by testing alternative data transformations that might be responsible for the data’s present state. This information then enables suitable accommodations to be made to relate the pattern instance to known pattern classes.

In order to demonstrate the validity of this approach, in chapter 5, DRNNM systems for Digit and Human Face Recognition were developed. The results from these developments were encouraging and provided a positive indication of the plausibility of the DRNNM for the recognition of dynamic data from minimal data examples. These DRNNM systems were contrasted with some existing static based approaches, furthering justifying this approach; the results indicating that under test conditions the DRNNM’s performance was significantly better.

6.2 Discussion

Based on this research it is possible to come to a number of conclusions regarding dynamic pattern recognition with scarce static resources. These include:

a) criticisms of existing classifier designs. These approaches are limited because they:

- learn solely by drawing on associations based on the data values at different data locations.
- have no means of accessing, or understanding, the data’s environment.
• fail to appreciate that recognition is a dynamic process

• fail to realise that recognition is dependent upon a knowledge of the data’s
dynamics, and that this knowledge cannot be gained by data correlation or
association.

b) the essential requirements needed to allow robust recognition under these
conditions:

• The separation of what is being recognised from the data representation.

• The encoding of knowledge as the capability to transform data, as opposed
to feature associations developed during training.

• The underlying concept that pattern data varies for a reason, and therefore
different class instances may, and can be, related in non-intuitive ways.

• The recognition of dynamic data requires a dynamic recognition process.

• The need for independent specialised modules to accommodate different
recognition stages

• The clear delineation between domain characteristics and recognition data.

6.3 Conclusions

The model developed in this thesis addressed these issues by:

• ensuring that it clearly defines what was, and has been learnt, by changing
the learning emphasis from class association to class dynamics.

• removing the difficulty in determining how well the system has learnt by
defining definite learning objectives

• simplifying training through the reduction of unnecessary class associations
• providing a set of behavioural indicators that are indicative of training and recognition problems

• designing the system to perform dynamically

• developing a recognition framework through which the system can interact with the recognition data without affecting the data integrity.

• clearly defining definite stages in the recognition process.

• developing different modules that incorporate the functionality of each recognition stage.

On the basis of this research, it is possible to conclude that the DRNNM provides an alternative approach to resolving complex recognition problems with minimal data. Systems based on this model reduce the environmental dependency of the classifier and thereby allow the system greater access to the underlying pattern they are trying to learn. This removes the emphasis in learning from class associations to class dynamics, thereby reducing learning complexity leading to a more modular system. This modularity, in turn, enables clear determination of system classification problems and direct access to the system’s knowledge. This allows the defining of specific learning objectives and a means of determining the success, or failure, of each learning task. This is reflected dynamically through the use of behavioural indicators that provide an assessment of the system’s progress within a recognition cycle.
The DRNNM provides a total recognition framework that can be readapted and applied to virtually any recognition problem. The DRNNM operates by initially defining a recognition domain’s limits, and then providing a means of determining the presence or absence of different domain characteristics or states. The more detailed this information the greater the potential success of the model within that domain. This differs from existing systems in that usually the training data is used to define the domain limitations. However, when resources are scarce this option is inappropriate. Consequently by defining the domain dynamics the DRNNM avoids a significant limitation of existing systems. The DRNNM is therefore able to overcome the static limitations of existing models.

A disadvantage of the DRNNM is its relative complexity. The model consists of seven distinctive modules that need to all interact for effective recognition. This can be problematic as the requirements of each module are different and need to be individually monitored. Therefore, the system can be potentially, complex and cumbersome to use when compared with an equivalent static recognition system.
Bibliography


Kellman, P. J., Spelke, E. S. “Perception of partly occluded objects in infancy”, *Cognitive Psychology*, 15, 483-524.


Tsotsos, John K. “Analyzing vision at the complexity level”, Behavioural and Brain Sciences(1990), 13, 423-469.


Appendix A : Supplemental Digit Results

A.1 Digit 3

Figure A.1 Normalisation of Digit 3

185
Digit 3-5 decision: translate digit to the right

Digit 3-6 decision: translate digit to the right

Digit 3-6 decision: translate digit down

Digit 3-7 decision: translate digit to the left

Digit 3-8 decision: rotate digit to the right

Figure A. 2 Classifier Responses for Transformation of Digit 3
Recognition (Digit 3)

Figure A.3 Holistic Digit Recognition Results for Digit 3

Figure A.4 MLP-Based Digit Recognition Results for Digit 3
Figure A.5 DRNNM Recognition Results for Digit 3
A.2 Digit 4.

Figure A.6 Normalisation of Digit 4

Digit 4-1 decision: translate digit to the right.

Digit 4-3 decision: rotate left

Digit 4-2 decision: translate digit down right

Digit 4-4 decision: translate digit left
Figure A.7 Classifier Responses for Transformation of Digit 4

Recognition Results (Digit 4)

Figure A.8 Holistic Digit Recognition Results for Digit 4
Figure A.9 MLP-Based Digit Recognition Results for Digit 4

Figure A.10 DRNNM Digit Recognition Results for Digit 4

A.3 Digit 5.
Figure A.11 Normalisation of Digit 5

Digit 5-1 **decision**: rotate digit left

Digit 5-2 **decision**: translate digit up-left

Digit 5-3 **decision**: translate digit left

Digit 5-4 **decision**: translate digit up

Digit 5-5 **decision**: translate digit right

Digit 5-6 **decision**: down-right
Digit 5-7 decision: translate digit left

Figure A.12 Classifier Responses for Transformation of Digit 5
Recognition (Digit 5)

Digit 5-8 attempt recognition

Figure A.13 Holistic Digit Recognition Results for Digit 5
The DRNNM had considerable difficulty in normalising this digit; almost entering a recognition cycle. This example clearly demonstrates the effect of the binding problem, in that only a single translation was the difference between a lackluster and a reasonably confident response.
A.4 Digit 6.

![Digit 6 images]

selected =

Figure A.16 Normalisation of Digit 6

![Graphs showing normalisation of Digit 6]

Digit 6-1 decision: up-right

Digit 6-2 decision: left

Digit 6-3 decision: rotate left

Digit 6-4 decision: Confidence is not improving - attempt recognition

Figure A.17 Classifier Responses for Transformation of Digit 6
Recognition (Digit 6)

Figure A.18 Holistic Digit Recognition Results for Digit 6

Figure A.19 MLP-Based Digit Recognition Results for Digit 6
Figure A.20 DRNNM Digit Recognition Results for Digit 6
A.5 Digit 7.

Figure A.21 Normalisation of Digit 7

Figure A.21 Classifier Responses for Transformation of Digit-7 Recognition (Digit 7)

Figure A.22 MLP-Based Digit Recognition Results for Digit 7
Figure A.23 DRNNM Digit Recognition Results for Digit 7
A.6 Digit 8.

Figure A.24 Normalisation of Digit 8

Digit 8-1 decision: rotate digit left

Digit 8-2 decision: translate digit right

Digit 8-3 decision: translate digit right

Digit 8-4 decision: translate digit right
Digit 8-5 decision: rotate digit left

Digit 8-6 decision: translate digit left down

Digit 8-7 decision: scale digit down - not making any progress terminate recognition

Figure A.25 Classifier Responses for Transformation of Digit-8
Recognition (Digit 8)

Figure A.26 Holistic Digit Recognition Results for Digit 8

Figure A.27 MLP-Based Digit Recognition Results for Digit 8
Figure A.28 DRNNM Digit Recognition Results for Digit 8
A.7 Digit 9.

Figure A.29 Normalisation of Digit 9

Digit 9-1 decision: translate digit down

Digit 9-2 decision: translate digit left

Digit 9-3 decision: translate digit up right

Digit 9-4 decision: rotate digit right

Figure A.30 Classifier Responses for Transformation of Digit-9
Recognition (Digit 9)

Figure A.31 Holistic Digit Recognition Results for Digit 9

Figure A.32 MLP-Based Digit Recognition Results for Digit 9
Figure A.33 DRNNM Digit Recognition Results for Digit 9
Appendix B : Related Conference Papers


