Classifying images using goal-seeking neural network architectures

A. de Carvalho
M.C. Fairhurst
D.L. Bisset

Indexing terms: Neural networks, Image classification

Abstract: New results, which have been achieved by using neural network architectures for two-dimensional image classification based on the goal-seeking neuron (GSN), are presented. A number of important practical issues concerning mapping topologies and the parallel implementation of GSN-based architectures are also investigated, together with a proposal for the development of a related neurally based feature extractor to be used as a front-end processor in a fully integrated Boolean network architecture.

1 Introduction

Neural network approaches to image classification have become increasingly widespread in recent years, but have often been somewhat unsatisfactory for many practical applications because of the constraints imposed on the processing speeds attainable, particularly in the implementation of appropriate training algorithms. Boolean neural networks offer an alternative approach to neural network design. Boolean neural architectures are also often referred to as RAM-based architectures, since their neurons can be implemented by RAM devices, and this feature, coupled with the fact that they usually input and output binary values, makes their hardware implementation potentially very simple and efficient. A further advantage of Boolean neural networks is that they can generally use fast (often single-shot) learning algorithms, and the high functionality of individual Boolean neurons offers a large degree of processing flexibility.

2 The GSN neuron model

Recent work has developed an approach to neural network design based on the idea of a goal-seeking neuron (GSN) [1]. The GSN is a deterministic Boolean neuron designed with the aim of overcoming many of the problems potentially found in some earlier Boolean models, such as poor generalisation, inefficient memory storage, crosstalk and indeterminism. GSN networks differ from other Boolean networks in several important aspects. The most fundamental difference is in the use of an 'undefined' value u, in addition to the binary values. A GSN neuron (Fig. 1) can input, store and generate values

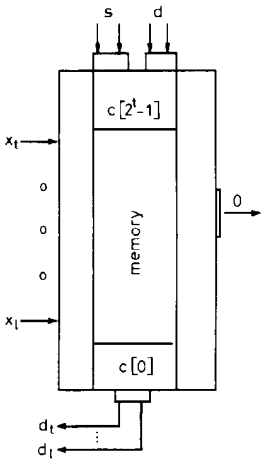

Fig. 1 *GSN neuron*
t = connectivity
x_i = input terminal to receive input pattern
$c_{[i]}$ = memory contents
o = output terminal to send the output of the neuron
d = input terminal to receive desired output
d_i = output terminal to send desired output to a neuron in the previous layer
s = learning strategy

equal to 0, 1 or u. If there is at least one value u on its input terminals, a set of memory contents will be addressed, the addresses of these memory locations being derived from the original address by changing the u values to 0 and 1. Before starting to train the network, all of its memory contents must be filled with undefined values, signifying that the network does not initially store any specific knowledge of the external world. By outputting undefined values, GSN allows them to be propagated in multilayer architectures, providing very good generalisation capabilities.

Similarly, GSN uses a much faster and more efficient learning algorithm. It is efficient because it maximises the efficiency of the storage of values in its memory contents, avoiding the storage of new information whenever possible and storing new information without disruption of previously stored information. It is fast because it employs one-shot learning algorithms, which means that the training set is presented just once (i.e. the training happens in a single epoch).

GSN neurons are usually grouped in pyramid structures. A pyramid is a multilayer structure where each layer has a fixed number of GSN Boolean neurons with a fan-out of 1 and a low fan-in (2 in the experiments reported here). The use of pyramids reduces the size of

Paper 9303I (C4, E4), first received 8th July and in revised form 28th October 1992

The authors are with the Electronic Engineering Laboratories, University of Kent, Canterbury, Kent CT2 7NT, United Kingdom

memory needed, increases generalisation by progressively reducing the number of different bits of similar input patterns as they are propagated through the successive layers, and makes the parallel implementation of GSN much easier. GSN neurons have been used in three different architectures: feedforward, feedback and self-organising, where in each architecture a different goal is associated with the neuron operation [1]. The feedforward and self-organising architectures are of primary interest here, and will be investigated in relation to their capabilities in the classification of two-dimensional images in a character-recognition task.

3 Feedforward structure

The most common GSN-based architecture is the GSN feedforward architecture (GSN^{ff}) [1]. As can be seen in Fig. 2, GSN^{ff} is formed by a fixed number of independent pyramids, where each pyramid covers a subset of the pixels of the input image. The choice of these subsets plays an important role in the task distribution among the pyramids, since a nonoptimal allocation can result in an inappropriate distribution, overloading certain pyramids and underloading others. Section 6 describes two different ways of selecting this subset.

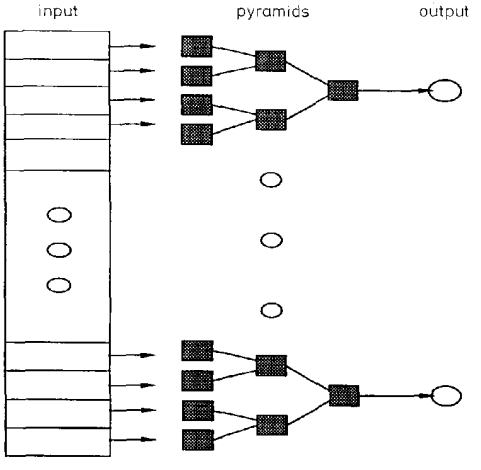

Fig. 2 *GSN^{ff} architecture*

The processing of the GSN^{ff} network can be divided into three phases, each associated with a specific goal: a validating phase, a learning phase and a recall phase. The purpose of the validating phase is to produce a validating value for each pyramid. This is the output value required at any pyramid in order for that pyramid to learn a given input pattern, and this establishes defined value(s) which can be learned in the next learning phase. A pyramid cannot be appropriately taught when its validating value does not correspond to its desired output value. The learning phase subsequently teaches the pyramids by appropriately changing the values stored in the memory locations of their neurons, to establish a desired output in response to a training pattern. The recall phase is responsible for classification of an input pattern, and seeks to output the defined value with the greatest occurrence in the addressable memory contents of the neuron at the apex of each pyramid.

In applying this architecture to a particular pattern recognition problem, a binary desired output vector is associated with each class. These vectors are defined before the learning takes place, and are used in both the learning and the recall phases. The desired output vectors are usually randomly defined, and this implies that, owing to the variability of the training set, the same pyramid may be trained with two different desired outputs for the same subpattern. By applying an algorithm to define the desired output based on the similarities among subpatterns of different classes and dissimilarities among subpatterns of the same class, an improvement in classification performance can generally be achieved.

Experimental investigations show that the learning algorithm used by this architecture makes performance very much dependent on the order of presentation of the patterns. A variation of as much as 4% in the recognition performance in a typical character recognition task can occur just by changing the sequence in which the training patterns are presented. In an attempt to reduce this influence, a new learning strategy has been developed where the desired output vectors are dynamically defined during the learning phase. This configuration is referred to as a quasi self-organising architecture.

3.1 Quasi self-organising GSN

The use of supervised learning in pyramid architectures presents a problem related to learning conflicts during the learning phase. A learning conflict occurs when the value generated by the validating phase for a given pyramid is the complement of the desired output value. As a consequence, that particular pyramid cannot learn the mapping between its input pattern and the desired output. The GSN feedforward architecture with quasi self-organising properties (GSN^{fq}) offers a strategy for dealing with this.

GSN^{fq} uses the same structure as that used by GSN^{ff}, but a different learning algorithm. In this algorithm the desired output can be modified when a conflict occurs and a pyramid is not able to learn the required association. The self-organising capabilities of this approach arise from the use of the desired output as a guess of the value to be learned by the pyramid. Depending on whether the 'rejected' element in the desired output vector is rejected frequently or not, its value van be changed dynamically during the learning phase. A statistical matrix is kept for all elements of all desired output vectors, and this is used to define the desired output vector each time a new pattern is presented during the learning phase. This matrix is built by counting, for each pyramid, the number of times each defined value is produced for each class during the validating phase. The new desired output vectors are then defined as

$$d_{ij} = \frac{n_{ij}^1}{n_{ij}^0 + n_{ij}^1} \tag{1}$$

where d_{ij} is the desired output of the pyramid i for the class j, and n_{ij}^0 and n_{ij}^1 are the number of times the validating values 0 and 1 are generated by the pyramid i for the class j.

4 Self-organising GSN

One of the most potentially useful features of neural networks is their capability for learning by example, without the necessity for being explicitly programmed. GSN^s is a self-organising Boolean architecture derived from the

GSN models described above, and is particularly applicable to pattern recognition problems.

As can be seen from Fig. 3, only one layer of neurons is now used, with the neurons being grouped in clusters

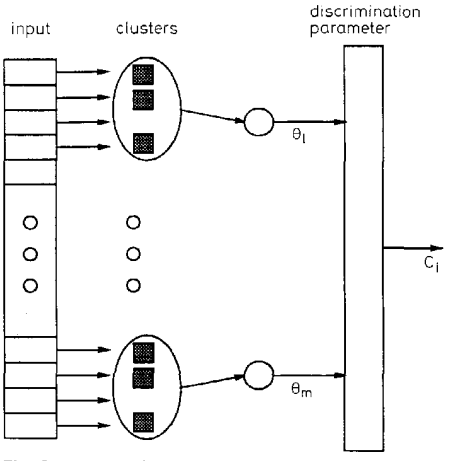

Fig. 3 *GSN^s architecture*

instead of pyramids. A cluster C_i can be thought of as the front layer of a pyramid P_i. GSN^s works by creating new clusters to store unfamiliar patterns, while grouping similar patterns in the same cluster. The computation performed by the GSN^s architecture is divided into two phases: the learning phase and the recognition phase. Both these phases use a match operation based on a function that measures the similarity between a new pattern and the coded representation of patterns stored in each one of the existent clusters [2].

During the training of this network, patterns belonging to the training data are coded and stored in the clusters. When the learning process is finished, each cluster is associated with one, and only one, class of pattern, although the same class can be represented by more than one cluster. In the recognition phase the match values of all the clusters are defined. As each class can be represented by more than one cluster, only the cluster with the maximum match value from each class contributes to the recognition decision. The pattern is then recognised as belonging to the class associated with the cluster with the biggest match value.

5 Experiments

To evaluate the performance achieved by these architectures in an image classification task, experimentation has been carried out on the recognition of machine-printed postcode characters. These characters have been extracted from real envelopes in the mail, and each is digitised as a 16 × 24 matrix. Although the data used here consist specifically of printed character samples, the quality is highly variable (Fig. 4) and the classification problem is correspondingly challenging. In these experiments, each network is trained with 100 samples extracted from 10 classes (those corresponding to the printed numerals 0–9), 10 samples per class. In testing, an independent set of 100 samples per class was used, giving a total of 1000 test patterns.

Both the GSN^{ff} and GSN^{fq} architectures adopted in these experiments used 400 pyramids with five neuron layers, each neuron having a fan-in equal to 2. The GSN^s structure initially comprises a single cluster, the learning process subsequently generating 24 clusters, each composed of 192 neurons. The performance of the GSN^{ff} architecture in this task is shown in Fig. 5. This graph shows how the rejection and correct classification rates are affected by increasing the rejection threshold.

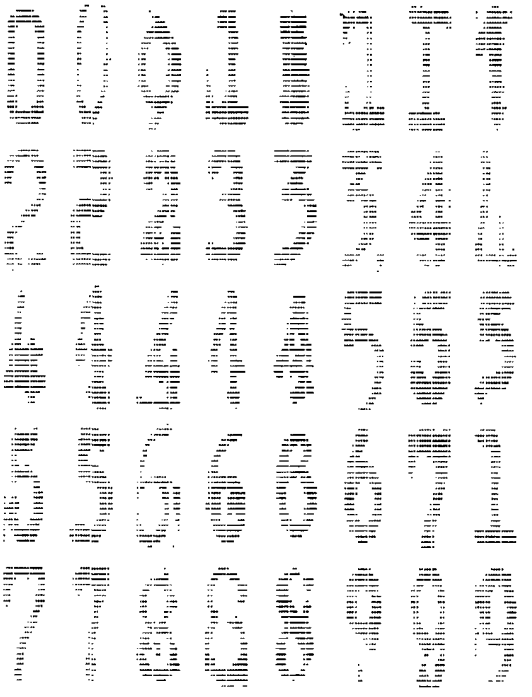

Fig. 4 *Typical samples of the characters used*

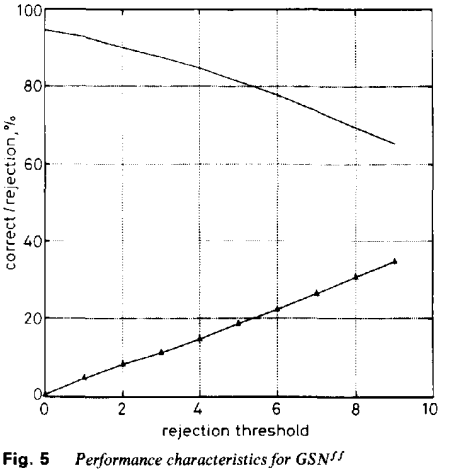

Fig. 5 *Performance characteristics for GSN^{ff}*
correct
rejection

When the same experiments are carried out with the alternative GSN^{fq} architecture, the corresponding results illustrated in Fig. 6 are obtained. These show that the dynamically defined desired output has improved the

recognition achieved by GSN^{ff}. Finally, Fig. 7 shows the results achieved using the GSN^{s} architecture.

Although the recognition performance achieved by GSN^{s} is broadly similar to the performance obtained using GSN^{fq}, the use of self-organising network offers a

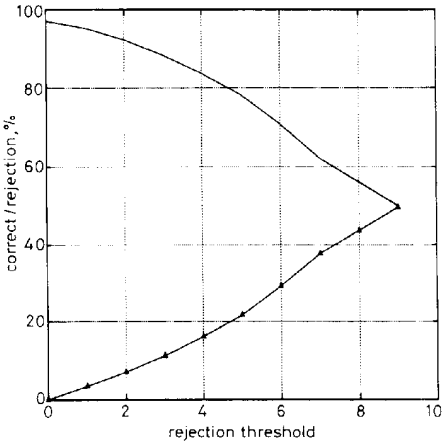

Fig. 6 *Performance characteristics for GSN^{fq}*
correct
rejection

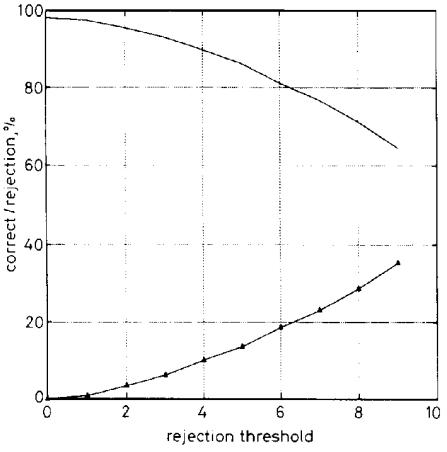

Fig. 7 *Performance characteristics for GSN^{s}*
correct
rejection

number of additional advantages. It does not need an external teacher to guide its learning process; it does not need to know in advance the number of classes among which the patterns might be distributed; and, generally, it requires much less memory in implementation (typically around 1200 bytes compared with 16 000 bytes needed by the GSN^{ff} architecture).

6 Mapping input terminals

An important step in building a GSN-based architecture is to define the way in which the input terminals of its neurons are connected. The choice of the connection map will define the functions that can be learned by each pyramid. The mapping is particularly important when using GSN^{ff} architectures, as each pyramid covers only a subarea of the whole input image.

The influence of different mappings, between the connections of the neurons in the first layer and the input, on the performance achieved by GSN^{ff} architectures, has been evaluated experimentally. Specifically, it is instructive to consider two alternative mapping schemes relating to two different fundamental structural strategies for defining connectivities. In the first mapping scheme, designated a 'continuous' mapping, the input area is divided into N blocks of equal size, one for each pyramid, where a block represents a set of pixels covering a continuous subarea of the input. In a second mapping scheme, designated a 'random' mapping, the input terminals of each pyramid are randomly connected to any pixel of the input image (Figs. 8 and 9).

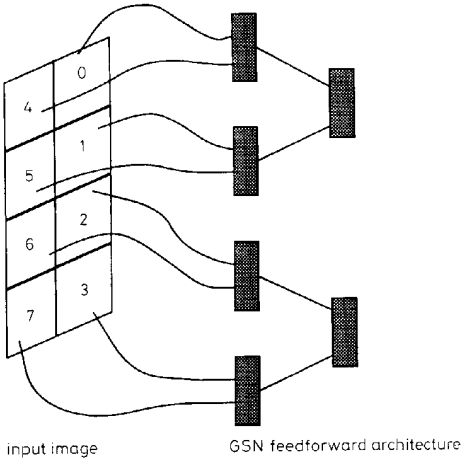

Fig. 8 *Continuous mapping*

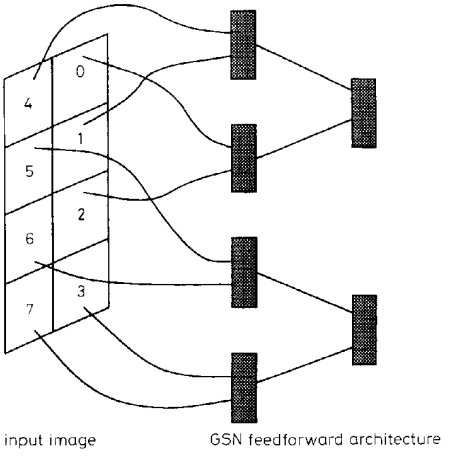

Fig. 9 *Random mapping*

Experiments were carried out to evaluate the influence of these two mappings on the performance of GSN^{ff} neural networks. The results achieved demonstrate that, with the continuous mapping, the distribution of the number of learning conflicts among the pyramids is less uniform. The reason is that each continuous subarea of the input image can involve a different degree of com-

plexity and, as all the pyramids are identical in terms of number of layers and connectivities, pyramids that cover more complex subareas will have to learn more complex associations between input subpatterns and their related desired output values. The differences in complexity among the continuous subareas is due to the fact that some subareas accommodate a larger diversity of subpatterns than others.

The precise influence of the mapping in the distribution of functions among the pyramids can be seen in Figs. 10 and 11, which show the number of conflicts occurring

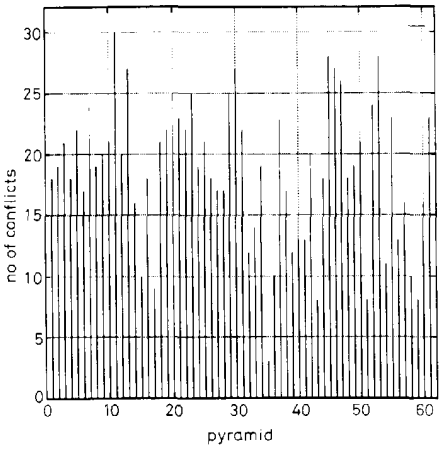

Fig. 10 *Continuous mapping*
—— conflicts

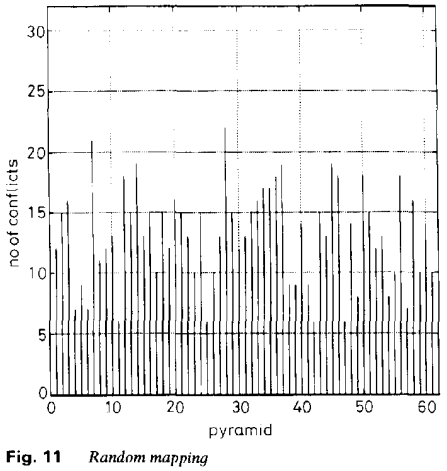

Fig. 11 *Random mapping*
—— conflicts

during learning for each pyramid when the two mapping approaches are used. It can be clearly seen that the number of conflicts is reduced, and that the differences in the number of conflicts across the pyramids is decreased when a random mapping is used. The number of conflicts is directly related to the complexity of the functions to be learned, as the more complex the functions, the more conflicts occur during learning and the fewer mappings that can be correctly learned. The greater irregularity in the distribution of the number of conflicts among the pyramids in the continuous case confirms a more unbalanced distribution of tasks among them.

Besides the irregular distribution of tasks among the pyramids, by restricting the mapping to pixels belonging to the same subarea the continuous mapping fails to capture important topological relations among pixels belonging to distinct subareas. Finally, the direct influence of the different mapping schemes on the recognition performance of the network was also investigated, and the results achieved using 10 different random mappings show that its adoption can result in an improvement of up to 8% in the recognition rate.

7 Parallel implementation

Due to the real-time requirements of many pattern recognition applications, a great deal of research effort has been devoted to the issue of implementing pattern recognition algorithms on parallel machines [3–5]. Transputer technology provides a very suitable environment for the implementation of massively parallel systems. Transputer hardware makes it possible to connect any number of processors in several different topologies, without the need for rigid synchronisation, and the structure and processing characteristics of GSN-based neural networks make them very suitable for massive parallel implementation. The independence of the constituent elemental structures in such networks — pyramids in the case of GSN^{ff} networks and clusters in GSN self-organising networks — and the low connectivity of GSN neurons makes them particularly suited to implementation using transputer networks. Both pyramids and clusters can be processed completely in parallel, with practically no interaction among them.

The question of parallel implementation has been investigated, and GSN^{ff} architectures have been successfully implemented on a network of transputers using different topologies, leading to a considerable improvement in processing speeds with respect to equivalent simulation on a serial machine. For example, a GSN^{ff} network with 432 pyramids of 31 nodes can be trained in 8 s when implemented on an array of 13 transputers connected in a tree topology, compared with 100 s when the implementation uses a single transputer; the classification times are 10 patterns per second and one pattern per second respectively. Such performance figures demonstrate a capacity for overall processing times attainable with a parallel implementation which are quite acceptable for a wide range of practical applications.

In the implementation of this network using 13 transputers, the 432 pyramids were equally distributed among 12 'worker' processors. The 13th processor, the 'master', was used to support the network, being responsible for operations such as input/output, classification and generation of parameter values. The topology used was a conventional tree topology, where each nonleaf transputer has three children. The master processor is the root of the tree.

8 GSN feature extraction

One of the potential difficulties with the structures discussed here is that the 'features' used in classification are essentially unstructured. A possible way to improve further the recognition performance of GSN-based architectures is to provide a preprocessing stage between the input patterns and the classification network. This preprocessor, acting as a feature extractor, would reduce the

network processing complexity by providing it with a set of individual primitive features, instead of the whole pattern presented as an unstructured entity. A GSN self-organising network can, in principle, be used for just this purpose, and this idea has been explored experimentally

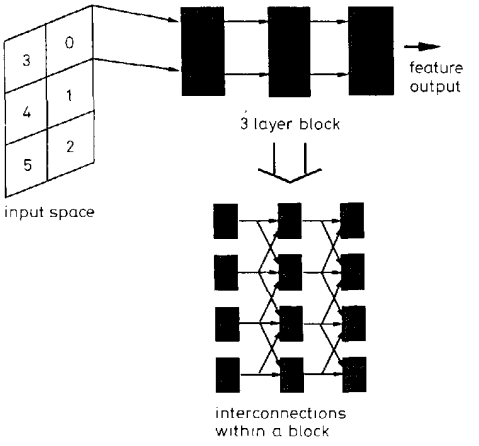

Fig. 12 *Feature extractor architecture*

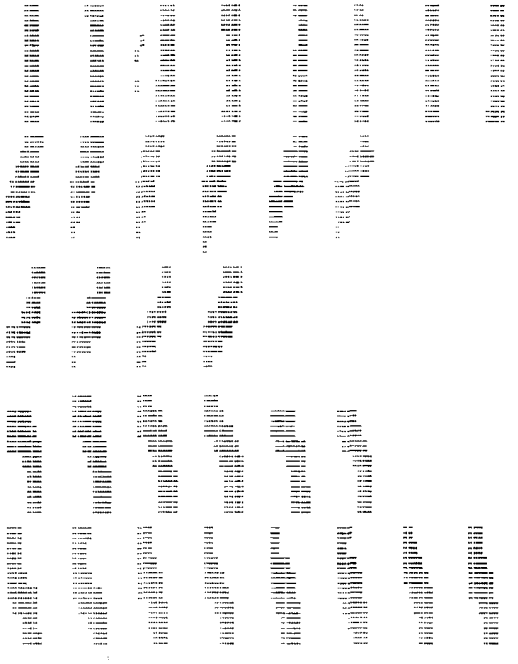

Fig. 13 *Samples of features extracted*

as the first step in progressing towards a more efficient and integrated neurally based recognition system.

In this architecture the neurons are grouped in a set of blocks, where each block is a lattice structure. The input image is divided into six areas, with a block being associated with each area (Fig. 12). By using an unsupervised learning algorithm, each block trains each of its neurons to distribute the input patterns between two groups, and, as a result, each block learns to discriminate the universe of input patterns present during the learning phase as a number of subsets, where each subset represents a particular feature. The learning phase associates each block with its own set of features. Each class to be recognised is then associated with a subset of features for each area, with each feature being weighted according to the frequency with which it occurs.

Preliminary results show that significant features are being extracted, and that classes whose patterns are similar in some of the six areas share some of their features in these areas. A curve in the upper left area, for example, appears with high frequency only for the classes 0, 2, 3, 6, 8 and 9. This allows postprocessing to remove infrequent features and improve the efficiency of the network structure.

An analysis of the output generated by this architecture shows that the features commonly detected are curves with different radii, strokes with diverse inclinations, and ends of curves and strokes. Fig. 13 shows a representative sample of input subpatterns associated with single features, where each row corresponds to a particular feature generated by a specific block. This gives a general idea of the kind of features that are extracted, and of the similarity of the subpatterns associated with the same feature.

These initial results suggest that the development of an integrated Boolean network architecture, encompassing both feature extraction and classification networks, is possible, and future work will seek to explore further this type of structure in a practical way.

9 Conclusions

A range of neural architectures based on Boolean processing nodes has been described, and the relative merits of these architectures in a practical task involving the reading of postcode characters extracted from envelopes in the mail have been investigated. At a more specific level, it has been shown that the pattern of connectivity between the input matrix and the first layer of processing cells can directly influence the degree of conflict between the values to be stored at specific addresses within the nodes for different training patterns. Attention to this aspect of network configuration can, in turn, result in a significant improvement in classification performance, and therefore constitutes an important issue in network optimisation in a practical classification situation. In relation to the implementation of the network, and its implications in relation to processing speed, it has been shown that the GSN architecture maps conveniently to a parallel infrastructure, and that training and classification times can be reduced by an order of magnitude for a typical network when implemented on even a modest array of transputer processors.

Finally, it is suggested that in many practical image classification tasks an important future direction will be to focus more specifically on the appropriate encoding of the raw image data through a feature extraction process, itself realised by means of a Boolean network. A framework for this type of task has been established, and it is clear that the potential for an integrated network exists.

The application of GSN-based neural networks to the practical problem of character reading has therefore been investigated in a way which establishes a basic framework, within which it is possible to tackle a range of similar practical tasks. Further work is focusing specifically on the development of network configurations for feature extraction and their integration with classifiers implemented using GSN networks, and as preprocessors

for more conventionally structured classification processes.

10 References

1 FILHO, E., FAIRHURST, M.C., and BISSET, D.L.: 'Adaptive pattern recognition using goal-seeking neurons', *Patt. Recog. Lett.*, March 1991, **12**, pp. 131–138

2 DE CARVALHO, A., FAIRHURST, M.C., BISSET, D.L., and FILHO, E.: 'An analysis of self-organising networks based on goal seeking neurons'. In Proc. Second IEE International Conference on Artificial Neural Networks, Bournemouth, UK, November 1991, pp. 257–261

3 MAHADEVAN, I., and PATNAIK, L.M.: 'Performance evaluation of bidirectional associative memory on a transputer-based parallel system', *Parallel Comput.*, 1992, **18**, pp. 401–413

4 OBERMAYER, K., RITTER, H., and SCHULTEN, K.: 'Large-scale simulations of self-organising neural networks on parallel computers: application to biological modelling', *Parallel Comput.*, 1990, **14**, pp. 381–404

5 YOON, H., NANG, J.H., and MAENG, S.R.: 'Parallel simulation of multilayered neural networks on distributed-memory multiprocessors', *Microproc. Microprog.*, 1990, **29**, pp. 185–195