Incorporating Scale Invariance into the Cellular Associative Neural Network

Nathan Burles, Simon O'Keefe, and James Austin

Department of Computer Science,
University of York,
York, YO10 5GH, UK
{nburles,sok,austin}@cs.york.ac.uk
http://www.cs.york.ac.uk

Abstract. This paper describes an improvement to the Cellular Associative Neural Network, an architecture based on the distributed model of a cellular automaton, allowing it to perform scale invariant pattern matching. The use of tensor products and superposition of patterns allows the system to recall patterns at multiple resolutions simultaneously. Our experimental results show that the architecture is capable of scale invariant pattern matching, but that further investigation is needed to reduce the distortion introduced by image scaling.

Keywords: pattern recognition, scale invariance, associative memory, correlation matrix memory, distributed computation, cellular automata

1 Introduction

Cellular automata are formed by connecting simple processing units, known as cells, into a grid or array. Although each individual cell may be simple, exchanging information with their neighbours to update their state can lead to complex or emergent behaviour from the cellular automata. Due to this ability, they are well suited to use for parallel and distributed processing [1].

Architectures based on cellular automata have been used successfully to solve low and medium level problems in computer vision [2]. They have also been extended to incorporate features from neural networks, with similar applications, using weight matrices and continuous time dynamics to replace the simple automaton rules [3]. Orovas introduced an alternative architecture which uses simple correlation matrix memories in each cell, in order to provide fast and efficient processing. The Cellular Associative Neural Network (CANN) was shown to be capable of distributed symbolic processing and pattern recognition with position invariance, but without scale invariance [4].

Apart from the most basic brute force technique—trying to match a pattern at numerous different scales—there are various ways in which scale invariance has been achieved in pattern recognition, using the detection of edges and interest points. These include the Generalised Hough Transform [5], graph matching [6], geometric hashing [7], or curvature scale space [8]. These use a range of analytical techniques, such as statistical and probabilistic models, and Gaussian filtering. None of these methods are suitable, however, for the distributed network of the CANN. Instead we will introduce a novel adaptation of the brute force technique, designed to minimise the performance penalty usually associated with it.

1.1 Correlation Matrix Memories (CMMs)

CMMs are simple, fully connected, associative neural networks consisting of a single layer of weights. Despite this simplicity they are still an active area of research and have been incorporated in a number of complex architectures, including the Associative Rule Chaining Architecture [9] and the Cellular Associative Neural Network. In this work we use binary CMMs, a sub-class of CMMs where the inputs, outputs, and weights are restricted to binary values [10].

Binary CMMs use simple Hebbian learning [11]. Associating pairs of binary vectors, and storing these associations within a CMM, is thus an efficient operation that requires only local updates to the CMM. Equation 1 formalises this learning, where $\mathbf{M}$ is the resulting CMM (or matrix of binary weights), $\mathbf{x}$ is the set of input vectors, $\mathbf{y}$ is the set of output vectors, n is the number of training pairs, and $\vee$ indicates the logical OR of binary vectors.

$$\mathbf{M} = \bigvee_{i=1}^{n} \mathbf{x}_i \mathbf{y}_i{}^T \tag{1}$$

To retrieve information from a CMM, a recall operation is performed as shown in Equation 2. A matrix multiplication between the transposed input vector $\mathbf{x}^T$ and the CMM $\mathbf{M}$ results in a non-binary output vector. A threshold function f must then be applied to this, in order to produce the final binary output vector.

$$\mathbf{y} = f(\mathbf{x}^T \mathbf{M}) \tag{2}$$

There are a number of functions which may be used as the threshold during a recall, although the choice of function may be limited by the application and the data representation used. In this application we use Willshaw's method of thresholding. We know the number of bits set to one in the input vectors during training. On recall, any output element with a value at least this large is set to one. Other output elements are set to zero [10]. This threshold function allows CMMs to operate correctly when superimposed vectors are presented for recall. Relaxation and partial matching is also simple to achieve, by reducing the threshold value.

2 The Cellular Associative Neural Network (CANN)

The CANN is an array of cells—known as associative processors—each of which contains a number of modules. The modules use CMMs to store rules that symbolically describe an object. Each module contains one or more CMMs, configured as an arity network,[1] in order that the number of antecedents to a rule can be variable [12].

[1] Arity networks use a number of CMMs, one for each possible arity, to store rules in order that a recall can be performed without unwanted partial matching

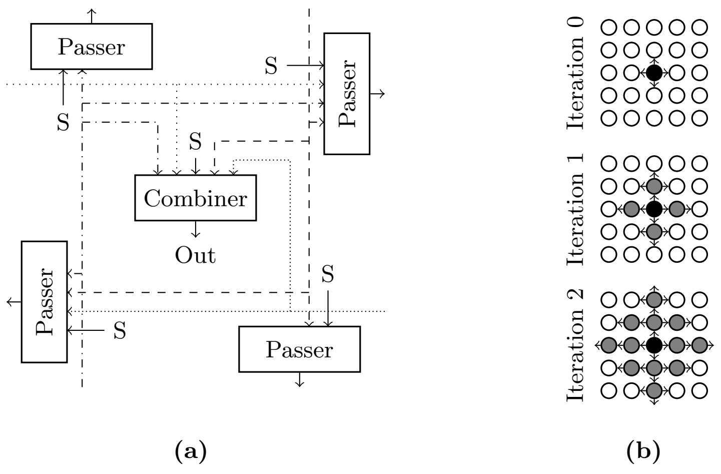

Fig. 1. (a) The "Corner Turning 2" CANN module configuration, where S is the cell state (the output of the combiner module) after each iteration and (b) information flow of this configuration over two iterations [13].

Learning and recognition of an object's structure uses a hierarchical approach, and cells exchange symbolic information with their four direct neighbours during each iteration. This means that after n iterations, each cell is made aware of the state of all cells up to a Manhattan distance of n from it. Various module configurations for the 2D CANN have been investigated, in order to optimise this message passing.

Brewer [13] showed that the "Corner Turning 2" configuration shown in Figure 1a provides the best performance of those tested—allowing information to travel between any two cells, while requiring fewer total rules than alternative suitable configurations. The data flow of this configuration is shown in Figure 1b, where the black cell is the origin of a piece of information, grey cells are those to which the information has been passed, and white cells are those which are not yet aware of the information.

2.1 Learning

Before learning an object's structure, a number of "primitives" are first recognised in the image—vertical and horizontal lines, and the four types of corner: |, —, ⌜, ⌝, ⌞, ⌟. Each of these primitives is then represented by a vector, and forms the initial input to each cell. During an iteration, information is received from each of a cell's neighbours' passer modules to form the antecedents of a rule. For each module, an input vector is created by appending the information received from neighbouring cells to the cells' input, according to the module configuration. This vector is then used to recall a new cell state from the combiner module, or a new information vector from a passer module. The module configuration determines the position that a vector is appended within the input, allowing a cell to distinguish between information received from different neighbours.

If a recall does not result in any output then this combination of antecedents has not been seen previously. In this case a new vector—a “transition symbol”—is generated to form the consequent of a rule, and associated with the antecedents into the relevant module. This transition symbol—whether it has been recalled or newly generated—is used either as the cell’s output state or to form the new information to be passed to the cell’s neighbours, depending on the module. When a transition symbol is generated, the new rule must be communicated to all other cells. This ensures that all cells contain the same information which allows the CANN to be translation invariant.

Finally, when every cell that initially contained a primitive has been assigned a unique state, the termination condition for learning an object is reached. At this point each of these cells generates a final rule to be stored in the combiner module. As with all other iterations the superimposed inputs are used as the antecedents, however the consequent in this case is a user provided symbol which denotes the learnt object.

Appending vectors in order to create a module input, rather than superimposing them, is suitable because each module has a fixed number of inputs n_i. This means that the module has a fixed input length of $n_i \times l$, where l is the vector length used in the system. If one of the cell’s neighbours does not have any information to pass, then an empty vector will be transferred and hence included in the input to one or more modules, leading to the requirement of an arity network. For example, in Figure 1a, the “combiner” module has a total of 5 inputs—all four neighbours, and the state of the cell itself (the output of the combiner module). If a vector weight of 4 is chosen, then when all the inputs contain information the total weight is $5 \times 4 = 20$—this is used as the value for Willshaw’s threshold. If one of the cell’s neighbours does not pass any information, however, then the total weight will only be $4 \times 4 = 16$. If this were to be recalled from a CMM with a threshold value of 20, then it could never result in an output. As such, it is stored in a separate CMM with a threshold value of 16. A recall operation can then present the input vector to the correct CMM in this arity network, using the relevant threshold value.

2.2 Recall

When a pattern is presented for recall, the operation of the CANN is similar to that of a cellular automaton. The rules which govern state transitions are stored in the various modules—with each cell containing exactly the same rules, to allow a pattern to be recognised by any group of cells. To begin a recall, the primitives are extracted from the pattern and used as the input to each cell. As with the learning process, a recall happens iteratively; during each iteration information is received from each of a cell’s neighbours and appended to its input, before recall from each of the modules.

If the pattern is recognisable, then after a number of iterations it is labelled with a symbol representing the object. It is unrealistic to expect a perfect recall to happen in every case, however, due to factors such as noisy inputs, distortion, and occlusion. In these cases, the system is able to generalise by taking advantage of a CMM's ability to perform partial matching. If, at any stage, a consequent is not successfully recalled from a module, then relaxation can be employed—that is to say that the threshold value will be reduced in order that an incomplete match may be attempted. This also allows the CANN to recognise inputs which are similar to patterns which have been previously trained [13].

3 Incorporating Scale Invariance

There are two obvious but impractical methods which may be used in order to incorporate scale invariance into the CANN in a neural and distributed manner, both a variant of the brute force method. The first requires training the CANN on multiple versions of the same pattern, presented at numerous different scales (within a predetermined range). This will increase the time required to initially train the network, but allow a recall to be performed quickly. Notably, however, it will significantly increase the number of rules generated—and hence the memory required to store these rules.

The second method trains only a single version of a pattern, presented at its original scale. A pattern must now be presented for recall at numerous different scales (within a predetermined range), in order that the CANN may find a match with the originally trained pattern. This minimises the number of rules generated and the memory required, however potentially imposes a great penalty on every recall performed.

We propose a novel third method which requires only a single version of each pattern to be trained, while minimising the performance penalty imposed by individually recalling a pattern at numerous different scales. Smolensky introduced the concept of a tensor product as a structure which stores bindings between variables and their values [14], and they have been widely investigated, e.g. [15]. Previously we showed that tensor products formed between input data and unique, randomly-generated, binary vectors may be superimposed and successfully recalled from a CMM [16]. Using this technique, we can improve upon the second method by presenting a pattern for recall at numerous different scales simultaneously.

When recalling a pattern, the whole image is first scaled to each of the desired sizes—each of these images is assigned a unique binding vector. Primitives are extracted from each of the images in turn, and a tensor product is formed for each cell by binding this primitive to the image's binding vector. All the tensor products for a given cell are then superimposed, and recall continues in the original fashion—in this case recalling each column of the tensor product in turn. If a pattern is recognised, it is possible to determine the scale at which it was found. Vectors remain in a tensor product throughout the operation of the system, which means that any assigned object labels are also in a tensor product. If this final tensor product is treated as a CMM, and the object label is presented as an input, then the output vector is the binding vector that was originally assigned to the scaled input pattern.

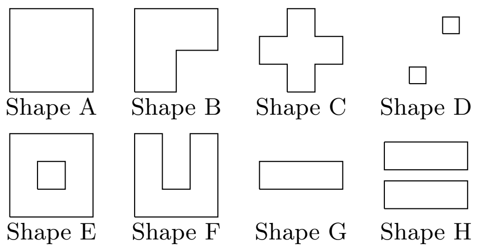

Fig. 2. The 8 patterns trained into the scale invariant CANN [13]

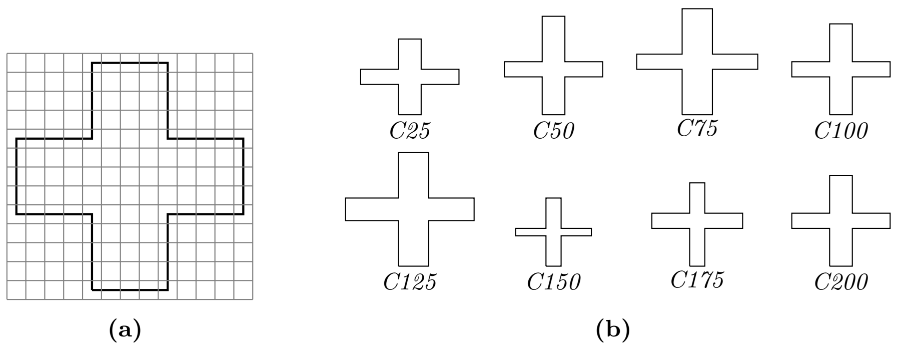

Fig. 3. (a) Cellular grid used when extracting primitives from Shape C and (b) the input image closest to 100% of the original size of Shape C, for each resized version from *C25* to *C200*.

3.1 Results

In order to test the recall success of the scale invariant CANN we trained the 8 patterns used in Brewer's previous work [13], shown in Figure 2, into a CANN using the original method. Each pattern was symbolically encoded by overlaying a grid, as shown for Shape C in Figure 3a, and extracting the primitive features to be used as the input for each cell.

Each of the symbolically encoded shapes was then presented for recall at a range of different sizes—every 25% between 25% and 200% of the original size. We then selected a range of scales to use when recalling, such that the scale invariant mechanism would not simply return the images to the original size and result in a perfect recall. Each input shape (e.g. *C25*) was scaled to a range of sizes—every 50% between 50% and 400% of its new size (e.g. *C75* would have been rescaled such that the superimposed recall input ranged from 37.5% to 300% of the original size of Shape C, in steps of 37.5%). As the shapes were already in symbolic form before resizing, the primitive features are immediately available to be bound to their respective binding vectors.

Figure 3b shows the input image closest to 100% of the original size of Shape C, for each resized version from *C25* to *C200*. Resizing the shapes when in symbolic form, rather than as images, has introduced significant variation and distortion. In future work using the scale invariant CANN, in order to achieve

Shape	**Scale of image presented for recall**							
	25%	**50%**	**75%**	**100%**	**125%**	**150%**	**175%**	**200%**
A	A 0.00	A 0.00	A 0.00	A 0.00	A 0.00	A 0.00	A 0.00	A 0.00
B	B 6.38	B 0.00	AB 44.83	B 0.00	AB 43.55	AB 27.78	B 0.00	B 0.00
C	C 0.00	C 7.69	C 43.33	C 0.00	100.00	C 10.53	C 0.00	C 0.00
D	D 0.00	D 0.00	D 0.00	D 0.00	D 0.00	D 0.00	D 0.00	D 0.00
E	E 3.45	E 0.00	E 61.11	E 0.00	E 43.59	E 22.73	E 3.57	E 0.00
F	F 0.00	F 0.00	AF 62.86	F 0.00	AF 60.81	F 13.64	F 0.00	F 0.00
G	G 0.00	A 80.00	A 80.95	G 0.00	A 82.61	G 0.00	G 12.50	G 0.00
H	H 0.00	H 0.00	AH 52.38	H 0.00	AH 56.52	H 0.00	H 6.25	H 0.00

Table 1. Error rates of the scale invariant CANN, when recalling Shapes A–H presented at scales ranging from 25% to 200%. Each result consists of the label(s) applied to the shape after recall, as well as the percentage of incorrectly labelled symbols.

the best results, images should be scaled before the primitive features are extracted. For this work, however, the variation serves as an important test of the CANN's ability to recognise distorted shapes.

Table 1 shows the results obtained when presenting the eight shapes for recall at each of the eight scales. Each result shows firstly the label or labels applied to the shape after recall, and then the percentage of the input shape which was incorrectly labelled. In the majority of cases, the shape was correctly labelled, however there are a number of errors that warrant further examination.

A number of recalled shapes, namely various scales of B, F, and H, were labelled as both A and the correct label. Similarly, three of the scales of Shape G were incorrectly labelled as Shape A. Given the similarities between these shapes, and the relaxation ability of the CANN, this is to be expected. This relaxation allows the CANN to recognise distorted and similar shapes, but can lead to incorrect recognition if two similar shapes are both initially trained into the CANN.

Presenting Shape C at a scale of 125% failed to result in any labels being applied. As can be seen in Figure 3b, the *C125* shape is the most distorted—being larger than at any other scale, as well as having different length arms. As mentioned earlier, this distortion could be reduced by scaling images before extracting the primitives.

4 Conclusions and Further Work

This paper has described an improvement to the CANN, to allow it to perform scale invariant pattern matching. Experimental results have shown that the architecture is capable of performing this task effectively, but that further work is needed to reduce the effect of the distortion introduced by image scaling.

The choice of which resolutions to use during a recall is very important, as this will affect how close to the original size a pattern may be scaled. The number of resolutions used may be increased, as the recall process happens simultaneously, however this may not be necessary if the distortion can be reduced by scaling images before extracting primitives. Further work is therefore required to determine the effect of this scaling, before attempting to determine whether there is an optimal set of resolutions to use (within a given range).

Finally, the CANN has largely been applied to synthetic patterns, although it has been shown to be able to operate successfully on simple photographic images [13]. Future work will further examine the application of the CANN to real objects in images.

References

1. Preston, K., Duff, M. (eds.): Modern Cellular Automata: Theory and Applications. Plenum Press (1984)
2. Saint-Pierre, de T., Milgram, M.: New and Efficient Cellular Algorithms for Image Processing. CVGIP: Image Understanding 55(3), 261–274 (1992)
3. Chua, L.O., Yang, L.: Cellular Neural Networks: Theory. IEEE Trans. on Circuits and Systems 35(10), 1257–1272 (1988)
4. Orovas, C., Austin, J.: Cellular Associative Neural Networks for Image Interpretation. In: Int. Conf. on Image Processing and its Appl., pp. 665–669. IET (1997)
5. Ballard, D.H.: Generalizing the Hough Transform to Detect Arbitrary Shapes. Pattern Recognition 13(2), 111–122 (1981)
6. Leung, T.K., Burl, M.C., Perona, P.: Finding Faces in Cluttered Scenes Using Random Labeled Graph Matching. In: Int. Conf. on Comput. Vis., pp. 637–644. IEEE (1995)
7. Wolfson, H.J., Rigoutsos, I.: Geometric Hashing: An Overview. Comput. Sci. & Eng. 4(4), 10–21 (1997)
8. Mokhtarian, F., Mackworth, A.K.: A Theory of Multiscale, Curvature-Based Shape Representation for Planar Curves. IEEE Trans. on Pattern Anal. and Mach. Intell. 14(8), 789–805 (1992)
9. Burles, N., O'Keefe, S., Austin, J.: Improving the Associative Rule Chaining Architecture. In: Artif. Neural Netw. and Mach. Learn.–ICANN, pp. 98–105. Springer (2013)
10. Willshaw, D.J., Buneman, O.P., Longuet-Higgins, H.C.: Non-holographic Associative Memory. Nature 222, 960–962 (1969)
11. Ritter, H., Martinetz, T., Schulten, K.: Neural Computation and Self-Organizing Maps: An Introduction. Addison-Wesley, Redwood City (1992)
12. Burles, N., Austin, J., O'Keefe, S.: Extending the Associative Rule Chaining Architecture for Multiple Arity Rules. In: Neural-Symb. Learn. and Reason., pp. 47–51. Beijing (2013)
13. Brewer, G.: Spiking Cellular Associative Neural Networks for Pattern Recognition. PhD Thesis, Univ. of York (2008)
14. Smolensky, P.: Tensor Product Variable Binding and the Representation of Symbolic Structures in Connectionist Systems. Artif. Intell. 46(1), 159–216 (1990)
15. Stewart, T., Eliasmith, C.: Compositionality and Biologically Plausible Models. In: Werning, M., Hinzen, W., Machery, E. (eds), The Oxf. Handb. of Compositionality. Oxford University Press, Oxford (2012)
16. Austin, J., Hobson, S., Burles, N., O'Keefe, S.: A Rule Chaining Architecture Using a Correlation Matrix Memory. In: Artif. Neural Netw. and Mach. Learn.–ICANN, pp. 49–56. Springer (2012)