Training Cellular Automata for Image Processing

Paul L. Rosin

Cardiff School of Computer Science,
Cardiff University, Cardiff, CF24 3AA, UK
Paul.Rosin@cs.cf.ac.uk,
http://users.cs.cf.ac.uk/Paul.Rosin

Abstract. Experiments were carried out to investigate the possibility of training cellular automata to to perform processing. Currently, only binary images are considered, but the space of rule sets is still very large. Various objective functions were considered, and sequential floating forward search used to select good rule sets for a range of tasks, namely: noise filtering, thinning, and convex hulls. Several modifications to the standard CA formulation were made (the B-rule and 2-cycle CAs) which were found to improve performance.

1 Introduction

Cellular automata (CA) consist of a regular grid of cells, each of which can be in only one of a finite number of possible states. The state of a cell is determined by the previous states of a surrounding neighbourhood of cells and is updated synchronously in discrete time steps. The identical rule contained in each cell is essentially a finite state machine, usually specified in the form of a rule table with an entry for every possible neighbourhood configuration of states.

Cellular automata are discrete dynamical systems, and they have been found useful for simulating and studying phenomena such as ordering, turbulence, chaos, symmetry-breaking, etc, and have had wide application in modelling systems in areas such as physics, biology, and sociology.

Over the last fifty years a variety of researchers (including well known names such as John von Neumann [15], Stephen Wolfram [16], and John Conway [7] have investigated the properties of cellular automata. Particularly in the 1960's and 1970's considerable effort was expended in developing special purpose hardware (e.g. CLIP) alongside developing rules for the application of the CAs to image analysis tasks [11]. More recently there has been a resurgence in interest in the properties of CAs without focusing on massively parallel hardware implementations. By the 1990's CAs could be applied to perform a range of computer vision tasks, such as: calculating properties of binary regions such as area, perimeter, convexity [5], gap filling and template matching [4], and noise filtering and sharpening [8],

One of the advantages of CAs is that, although each cell generally only contains a few simple rules, the combination of a matrix of cells with their local

H. Kalviainen et al. (Eds.): SCIA 2005, LNCS 3540, pp. 195–204, 2005.

interaction leads to more sophisticated emergent global behaviour. That is, although each cell has an extremely limited view of the system (just its immediate neighbours), localised information is propagated at each time step, enabling more global characteristics of the overall CA system.

A disadvantage with the CA systems described above is that the rules had to be carefully and laboriously generated by hand [14]. Not only is this tedious, but it does not scale well to larger problems. More recently there has been a start to automating rule generation using evolutionary algorithms. For instance, Sipper [13] shows results of evolving rules to perform thinning, and gap filling in isothetic rectangles. Although the tasks were fairly simple, and the results were only mediocre, his work demonstrates that the approach is feasible.

2 Design and Training of the CA

In the current experiments all input images are binary, and cells have two states (i.e. they represent white or black). Each cell's eight-way connected immediate neighbours are considered (i.e. the Moore neighbourhood). Fixed value boundary conditions are applied in which transition rules are only applied to non-boundary cells. The input image is provided as the initial cell values.

2.1 The Rule Set

Working with binary images means that all combinations of neighbour values gives 2^8 possible patterns or rules. Taking into account 90° rotational symmetry and bilateral reflection provides about a five-fold decrease in the number of rules, yielding 51 in total.

The 51 neighbourhood patterns are defined for a central black pixel, and the same patterns are inverted (i.e. black and white colours are swapped) for the equivalent rule corresponding to a central white pixel. According to the application there are several possibilities:

- both of these two central black and white rule sets can be learnt and applied separately,
- the two rule sets are considered equivalent, and each corresponding rule pair is either retained or rejected for use together, leading to a smaller search space of possible rule sets,
- just one of the black and white rule sets is appropriate, the other is ignored in training and application.

Examples of the latter two approaches will shown in the following sections.

2.2 Training Strategy

Most of the literature on cellular automata studies the effect of applying manually specified transition rules. The inverse problem of determining appropriate rules to produce a desired effect is hard [6]. In our case there are 2^{102} or 2^{51}

combinations of rules to be considered! Evolutionary methods appear to be preferred; for instance to solve the density classification task researchers have used genetic algorithms [10] and genetic programming [1]. Instead, we currently use a deterministic approach, which is essentially modified hill-climbing: the sequential floating forward search (SFFS) [12].

2.3 Objective Functions

An objective function is required to direct the SFFS, and various error measures have been considered in this paper. The first is root mean square (RMS) error between the input and target image.

In some applications there will be many more black pixels than white (or vice versa) and it may be preferable to quantify the errors of the black pixels separately from the white. This is done by computing the proportion B of black target pixels incorrectly coloured in the output image, and likewise W is computed for white target pixels. The combined error is taken as $B + W$.

The above measures do not consider the positions of pixels. In an attempt to incorporate spatial information, the distance at each incorrectly coloured pixel in the output image to the closest correctly coloured pixel in the target image is calculated. The final error is the summed distances. The distances can be determined efficiently using the distance transform of the target image.

A modification of the above is the Hausdorff distance. Rather than summing the distances only the maximum distance (error) is returned.

2.4 Extensions

There are many possible extensions to the basic CA mechanism described above. In this paper two modifications were implemented and tested. The first is based on Yu *et al.*'s [17] B-rule class of one dimensional CA. Each rule tests the value of the central pixel of the *previous* iteration in addition to the usual pixel and its neighbour's values at the *current* iteration. The second variation is to split up the application of the rules into two interleaved cycles. In the even numbered iterations one rule set is applied, and in the odd numbered iterations the other rule set is applied. The two rule sets are learnt using SFFS as before, and are not restricted to be disjoint.

3 Noise Filtering

The first experiment is on filtering to overcome salt and pepper noise. Figure 1 shows a portion of the large training and test images. Varying amounts of noise were added, and for each level the CA rules were learnt using the various strategies and evaluation criteria described above. In all instances the rules were run for 100 iterations. It was found that using the SFFS method with the RMS error criterion provided the best results, and unless otherwise stated all the results shown used this setup.

Table 1. RMS errors of filtered versions of single pixel salt and pepper noise

S & P prob.	orig.	median			CA	CA B-rule	CA 2 cycle
		1 iteration	100 iterations	optimal iterations			
0.01	320	1807	2925	1807 (1)	199	147	199
0.1	3247	2027	3065	2027 (1)	1048	1009	1044
0.3	9795	3113	3521	2836 (2)	2268	2272	2263

(a) (b)

Fig. 1. 585 × 475 sections of the 1536 × 1024 original images before noise was added; (a), (b) training/test images

(a) (b) (c) (d)

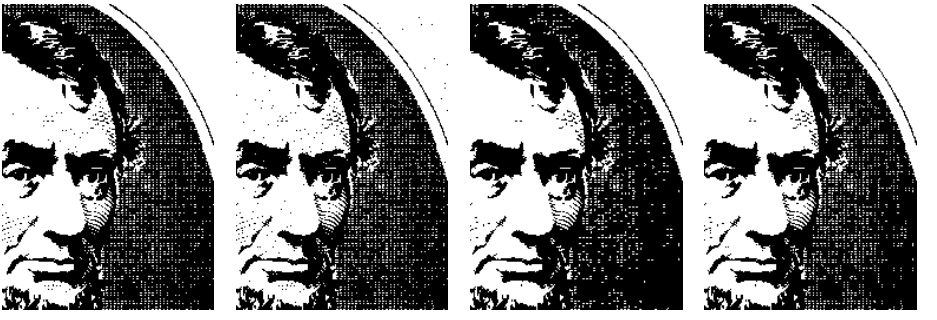
Fig. 2. Salt and pepper noise affecting single pixels occurring with a probability of 0.01; (a) unfiltered, (b) 1 iteration of median, (c) CA, (d) B-rule CA

For comparison, results of filtering with a 3 × 3 median filter are provided. While there are more sophisticated filters in the literature [3] this still provides a useful benchmark. Moreover, the optimal number of iterations of the median was determined for the *test* image, giving a favourable bias to the median results.

At low noise levels ($p = 0.01$) the CA learns to use a single rule[1] () to remove isolated pixels. As the RMS values show (table 1) this is considerably better than median filtering which in these conditions has its noise reduction overshadowed by the loss of detail, see figure 2. The B-rule CA produces even better results than the basic CA. 50 rules were learnt, although this is probably

[1] The rules are shown with a black central pixel – which is flipped after application of the rule. The neighbourhood pattern of eight white and/or black (displayed as gray) pixels which must be matched is shown. The rule set is shown (left to right) in the order that the rules were added to the set by the SFFS process.

far from a minimal set since most of then have little effect on the evaluation function during training. As before, the first rule is  , applied when the central pixel is a different colour in the previous iteration. In contrast, most of the remaining rules are applied when the central pixel is the same colour in the previous iteration. The difference in the outputs of the basic and B-rule CAs is most apparent on the finely patterned background to Lincoln (figure 3), which



(a) (b) (c) (d)

Fig. 3. An enlarged portion from the original and processed images with 0.01 probability salt and pepper noise. (a) original, (b) original with added noise, (c) filtered with CA, (d) filtered with CA B-rule

(a) (b) (c) (d)

Fig. 4. Salt and pepper noise affecting single pixels occurring with a probability of 0.1; (a) unfiltered, (b) 1 iteration of median, (c) CA, (d) B-rule CA

(a) (b) (c) (d)

Fig. 5. Salt and pepper noise affecting single pixels occurring with a probability of 0.3; (a) unfiltered, (b) 2 iterations of median, (c) CA, (d) B-rule CA

Table 2. RMS errors of filtered versions of 3×3 pixel salt and pepper noise

S & P prob.	orig.	3×3 median			5×5 median			CA	CA B-rule	CA 2 cycle
		1 iter.	100 iter.	opt. iter.	1 iter.	100 iter.	opt. iter.			
0.01	2819	3704	3465	3145 (3)	3923	6287	3923 (1)	2011	1425	1988
0.1	19886	18250	13583	13583 (39)	15621	9041	8930 (25)	8530	8090	8622

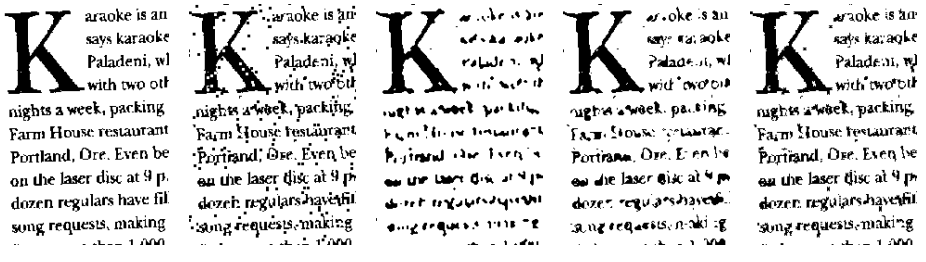
(a) (b) (c) (d) (e)

Fig. 6. Salt and pepper noise affecting 3×3 blocks occurring with a probability of 0.01; (a) unfiltered, (b) 3 iterations of median, (c) 1 iteration of 5×5 median, (d) CA, (e) B-rule CA

has been preserved while the noise on the face has still been removed. The 2-cycle CA produces identical results to the basic CA.

At greater noise levels the CA continues to perform consistently better than the median filter (figures 4 & 5). At $p = 0.1$ the learnt CA rule set is  and required 31 iterations for convergence. At $p = 0.3$ the learnt CA rule set is  and required 21 iterations for convergence. Again the 2-cycle CA produced little improvement over the basic CA, while the B-rule CA does at $p = 0.1$ but not $p = 0.3$. The B-rule rule sets are reasonably compact, and the one for $p = 0.1$ is shown: the rule set applied when the central pixel is a different colour in the previous iteration is  while for the same coloured central pixel at the previous iteration the rule set is  .

Increasing levels of noise obviously requires more filtering to restore the image. It is interesting to note that not only have more rules been selected as the



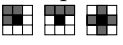
(a) (b) (c) (d) (e)

Fig. 7. An enlarged portion from the original and processed images with 0.01 probability salt and pepper noise. (a) original, (b) original with added noise, (c) filtered with 3 iterations of median filter, (d) filtered with CA, (e) filtered with CA B-rule

(a) (e) (g) (i) (i)

Fig. 8. Salt and pepper noise affecting 3×3 blocks occurring with a probability of 0.1; (a) unfiltered, (b) 39 iterations of median, (c) 25 iterations of 5×5 median, (d) CA, (e) B-rule CA

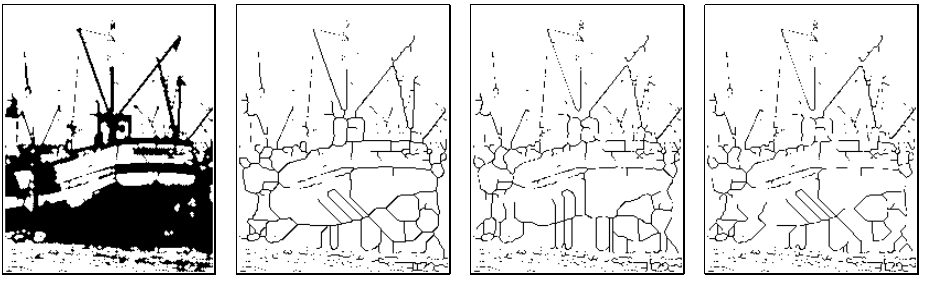
noise level increases, but also that, for the basic CA, they are strictly supersets of each other.

The second experiment makes the noise filtering more challenging by setting 3×3 blocks, rather than individual pixels, to black or white. However, the CA still operates on a 3×3 neighbourhood. Given the larger structure of the noise larger (5×5) median filters were used. However, at low noise levels ($p = 0.01$) the 3×3 median gave a lower RMS than the 5×5 although the later was better at high noise levels ($p = 0.1$). Nevertheless, the basic CA outperformed both (table 2). At $p = 0.01$ (figure 6) the learnt rule set was  and required 42 iterations for convergence. The B-rule CA further improved the result, and this can most clearly be seen in the fragment of text shown in figure 7. At $p = 0.1$ (figure 8) the learnt rule set was  and even after 100 iterations the CA had not converged. The 2-cycle CA did not show any consistent improvement over the basic CA.

4 Thinning

Training data was generated in two ways. First, some one pixel wide curves were taken as the target output, and were dilated by varying amounts to provide the pre-thinned input. In addition, some binary images were thinned by the thinning algorithm by Zhang and Suen [18]. Both sets of data were combined to form a composite training input and output image pair. Contrary to the image processing tasks in the previous sections the RMS criterion did not produce the best results, and instead the summed proportions of black pixel errors and white pixel errors was used. Surprisingly the summed distance and Hausdorff distance error measures gave very poor results. It had seemed likely that they would be more appropriate for this task given the sparse nature of skeletons which would lead to high error estimates for even small mislocations if spatial information were not incorporated. However, it was noted that they did not lead the SFFS procedure to a good solution. Both of them produced rule sets with higher errors than the rule set learnt using RMS, even according to their own measures.

The test image and target obtained by Zhang and Suen's thinning algorithm are shown in figures 9a&b. The basic CA does a reasonable job (figure 9c), and



(c) (d) (e) (f)

Fig. 9. Image thinning; (a) test input, (b) test image thinned using Zhang and Suen's algorithm, (c) test image thinned with CA, (d) test image thinned with 2 cycle CA

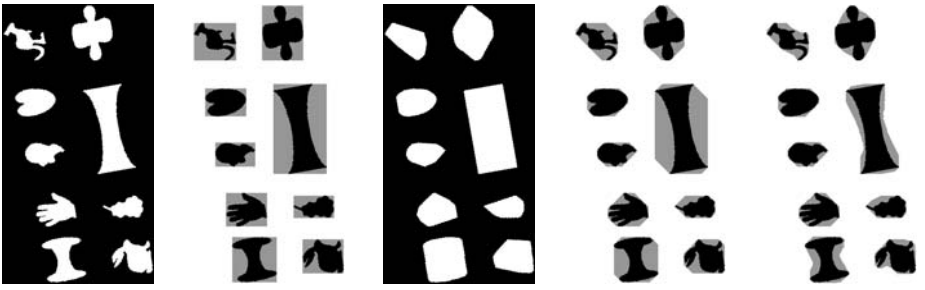
the rule set is . The last rule has little effect, only changing three pixels in the image. Some differences with respect to Zhang and Suen's output can be seen. In the wide black regions horizontal rather than diagonal skeletons are extracted, although it is not obvious which is more correct. Also, a more significant problem is that some lines were fragmented. This is not surprising since there are limitations when using parallel algorithms for thinning, as summarised by Lam *et al.* [9]. They state that to ensure connectedness either the neighbourhood needs to be larger than 3×3. Alternatively, 3×3 neighbourhoods can be used, but each iteration of application of the rules is divided into a series of subcycles in which different rules are applied.

This suggests that the two cycle CA should perform better. The rule set learnt for the first cycle is and the second cycle rule set is a subset of the first: . Again the last and least important rule from the first cycle has little effect (only changing 6 pixels) and so the basic CA and the first cycle of the B-rule have effectively the same rule set. As figure 9d shows, the output is a closer match to Zhang and Suen's, as the previously vertical skeleton segments are now diagonal, but connectivity is not improved.

5 Convex Hulls

The next experiment tackles finding the convex hulls of all regions in the image. If the regions are white then rules need only be applied at black pixels since white pixels should not be inverted. As for the thinning task, the summed proportions of black pixel errors and white pixel errors was used. After training the learnt rule set was applied to a separate test image (figure 10a). Starting with a simple approximation as the output target, a four-sided hull, i.e. the axis aligned minimum bounding rectangle (MBR), the CA is able to produce the correct result as shown in figure 10b. The rule set learnt is .

Setting as target the digitised true convex hull (see figure 10c) the CA learns to generate an eight-sided approximation to the convex hull (figure 10d) using



(a) (b) (c) (d) (e)

Fig. 10. Results of learning rules for the convex hull. (a) test input; (b) CA result with MBR as target overlaid on input; (c) target convex hull output; (d) CA result with (c) as target overlaid on input; (e) 2 cycle CA result with (c) as target overlaid on input

the rule set  . Interestingly, in comparison to the eight-sided output the only difference to the rules for the four-sided output is the removal of the single rule  . The limitations of the output convex hull are to be expected given the limitations of the current CA. Borgefors and Sanniti di Baja [2] describe parallel algorithms for approximating the convex hull of a pattern. Their 3×3 neighbourhood algorithm produces similar results to figure 10d. To produce better results they had to use larger neighbourhoods, and more complicated rules.

Therefore, extending the basic CAs capability by applying the 2-cycle version should enable the quality of the convex hull to be improved. As figure 10e shows the result is no longer convex although is is a closer match to the target in terms of its RMS error. This highlights the importance of the evaluation function. In this instance simply counting pixels is not sufficient, and a penalty function that avoids non-convex solutions would be preferable, although computationally more demanding.

6 Conclusions

The initial experiments with CAs are encouraging. It was shown that it is possible to learn good rule sets to perform common image processing tasks. Moreover, the modifications to the standard CA formulation (the B-rule and 2-cycle CAs) were found to improve performance. In particular, for filtering salt and pepper noise, the CA performed better than standard median filtering.

To further improve performance there are several areas to investigate. The first is alternative neighbourhood definitions (e.g. larger neighbourhoods, circular and other shaped neighbourhoods, different connectivity). Second, although several objective functions were evaluated, there may be better ones available – particularly if they are tuned to the specific image processing task. Third, can additional constraints be included to prune the search space, improving effi-

ciency and possibly effectiveness? Fourth, alternative training strategies to SFFS should be considered, such as evolutionary programming.

Most CAs use identical rules for each cell. An extension would be to use non-uniform CA, in which different rules could be applied at different locations, and possibly also at different time steps depending on local conditions.

References

1. D. Andre, F.H. Bennett III, and J.R. Koza. Discovery by genetic programming of a cellular automata rule that is better than any known rule for the majority classification problem. In *Proc. Genetic Prog.*, pages 3–11. MIT Press, 1996.
2. G. Borgefors and G. Sanniti di Baja. Analysing nonconvex 2D and 3D patterns. *Computer Vision and Image Understanding*, 63(1):145–157, 1996.
3. T. Chen and H.R. Wu. Application of partition-based median type filters for suppressing noise in images. *IEEE Trans. Image Proc.*, 10(6):829–836, 2001.
4. T. de Saint Pierre and M. Milgram. New and efficient cellular algorithms for image processing. *CVGIP: Image Understanding*, 55(3):261–274, 1992.
5. C.R. Dyer and A. Rosenfeld. Parallel image processing by memory-augmented cellular automata. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(1):29–41, 1981.
6. N. Ganguly, B.K. Sikdar, A. Deutsch, G. Canright, and P.P. Chaudhuri. A survey on cellular automata. Technical Report 9, Centre for High Performance Computing, Dresden University of Technology, 2003.
7. M. Gardner. The fantastic combinations of john conway's new solitaire game "life". *Scientific American*, pages 120–123, 1970.
8. G. Hernandez and H.J. Herrmann. Cellular automata for elementary image enhancement. *Graphical Models and Image Processing*, 58(1):82–89, 1996.
9. L. Lam and C.Y. Suen. An evaluation of parallel thinning algorithms for character-recognition. *IEEE Trans. PAMI*, 17(9):914–919, 1995.
10. F. Jiménez Morales, J.P. Crutchfield, and M. Mitchell. Evolving 2-d cellular automata to perform density classification. *Parallel Comp.*, 27:571–585, 2001.
11. K. Preston and M.J.B. Duff. *Modern Cellular Automata-Theory and Applications.* Plenum Press, 1984.
12. P. Pudil, J. Novovicova, and J.V. Kittler. Floating search methods in feature-selection. *Pattern Recognition Letters*, 15(11):1119–1125, 1994.
13. M. Sipper. The evolution of parallel cellular machines toward evolware. *BioSystems*, 42:29–43, 1997.
14. B. Viher, A. Dobnikar, and D. Zazula. Cellular automata and follicle recognition problem and possibilities of using cellular automata for image recognition purposes. *Int. J. of Medical Informatics*, 49(2):231–241, 1998.
15. J. von Neumann. *Theory of Self-Reproducing Automata.* University of Illinois Press, 1966.
16. S. Wolfram. *Cellular Automata and Complexity.* Addison-Wesley, 1994.
17. D. Yu, C. Ho, X. Yu, and S. Mori. On the application of cellular automata to image thinning with cellular neural network. In *Cellular Neural Networks and their Applications*, pages 210–215. 1992.
18. T.Y. Zhang and C.Y. Suen. A fast parallel algorithm for thinning digital patterns. *Comm. ACM*, 27(3):236–240, 1984.