Pergamon

Neural Networks, Vol. 10, No. 6, pp. 1143–1151, 1997
© 1997 Elsevier Science Ltd. All rights reserved
Printed in Great Britain
0893–6080/97 $17.00+.00

PII: S0893-6080(96)00128-1

CONTRIBUTED ARTICLE

Genetically Trained Cellular Neural Networks

MICHELE ZAMPARELLI

HLRZ-KFA 52425 , and Dipartimento di Matematica, Universitá degli Studi di Roma, "La Sapienza"

(*Received* 6 *June* 1995; *accepted* 10 *November* 1996)

Abstract—*Real-coded genetic algorithms on a parallel architecture are applied to optimize the synaptic couplings of a Cellular Neural Network for specific greyscale image processing tasks. Using supervised learning information in the fitness function, we propose the Genetic Algorithm as a general training method for Cellular Neural Networks.* © *1997 Elsevier Science Ltd.*

Keywords—Cellular neural networks, Genetic algorithms, Supervised learning, Image processing.

1. INTRODUCTION

Cellular Neural Networks (CNNs), a new type of locally connected neural network with continuous activation values, have recently demonstrated their efficacy for bipolar signal processing. Several models of cortical neurons have been proposed so far, but the time evolution in the neuron's activation has always been rather underestimated. CNNs on the other hand, have shown how the transient regime may play a decisive role in obtaining the correct stimulus–response association. An evolutionary approach, inspired from natural laws like the survival of the fittest, has already been proposed in several works, the main argument being that the actual brain structure has itself evolved through a competitive trial and error process. Genetic Algorithms (GAs), an attempt to emulate this trial and error process, have exhibited good performance in the design of feed-forward neural network achitectures (Bornholdt & Graudenz, 1992) and it is therefore quite natural to investigate their performance in modelling the synaptic connections which influence the dynamic trajectory of a neuron's activation level. In Sections 2 and 3 the notions of CNN and GA are briefly recalled. Section 4 describes the specific kind of GA used and Sections 5 and 6 give more details on the overall system and its parallelization. Some comments on the results are given in Section 7.

Acknowledgements: I would like to thank Eliano Pessa for his kind support and all the friends in HLRZ for the useful discussions. A special thanks to their boss, Hans Herrmann, for inviting me to this center.

All the simulations have been made on the KFA's Intel Paragon XP/S 10 with a variable number of processors. Financial support for this work came from E.E.C. Erasmus Programme.

Requests for reprints sould be sent to Michele Zamparelli, Institut für Theoretische Physik Humboldt-Universität zu Berlin, Invalidenstrasse 110, D-10099, Berlin; email: zamparelli@physik.hu-berlin.de.

2. THE CELLULAR NEURAL NETWORK MODEL

Cellular Neural Networks invented by Chua and Yang (1988), consist of a partial unification of the paradigms Cellular Automaton and Neural Network, retaining several elements of both. Neurons, intended as usual as elementary computational units, are placed on a regular two-dimensional lattice and are characterized by a *state* (**x**), *input* (**u**) and *output* (**y**). We will use the indices *i,j* for the case of a regular square lattice and we will denote with $N_{i,j}$ the neighbourhood of neuron *i,j*.

The state of a neuron is then ruled by the following first-order differential equation:

$$C\frac{dx_{ij}}{dt}(t) = -\frac{1}{R}x_{ij}(t) + \sum_{k,l\in N_{ij}} A_{kl}y_{kl}(t) + \sum_{k,l\in N_{ij}} B_{kl}u_{kl} + I \tag{1}$$

with the additional constraints

$$|u_{ij}| < 1,\ |x_{ij}(0)| < 1,\ C,R > 0 \tag{2}$$

and the choice of any non-linear, bounded, output function, such as

$$y_{ij} = \frac{1}{2}(|x_{ij}+1| - |x_{ij}-1|). \tag{3}$$

The matrices A_{kl} and B_{kl} are called, respectively, the *feedback* and *control templates* and represent the strength of the synaptic interaction between neighbouring neurons and the central neuron. Given a sufficient number of time steps, neurons not directly connected may interact with each other because of the propagation effects of the continuous-time dynamics of the network. This type of structure with only local synaptic connections makes CNNs ideally suited for VLSI implementation. In fact, CNNs were first conceived as a large scale non-linear circuit, with continuous range cells to overcome the limitations of discrete state Cellular Automata.

The neighbourhood radius r is typically chosen to be 1 or 2. C, R and I are respectively called the *capacity*, *resistance* and *induction* of the network. Eqn (1) is discretized through the corresponding difference equation

$$x_{i,j}(n+1)=x_{i,j}(n)+\frac{h}{C}$$

$$\left[-\frac{1}{R}x_{i,j}(n)+\sum_{k,l\in N_{ij}}A_{kl}y_{kl}(t)+\sum_{k,l\in N_{ij}}B_{kl}u_{kl}+I\right] \quad (4)$$

where h is the dynamic time step.

A CNN is uniquely determined given the feedback and control templates (A_{kl} and B_{kl}) and the additional three parameters C, R and I. It can be demonstrated that under conditions (2), the CNN will always converge to a steady state. The output is then used for evaluation.

The initial pattern representing the external stimulus may be loaded onto the initial state and/or onto the input. Several variations of the original CNN have been proposed and used for black and white image processing tasks, like edge detection, noise removal, horizontal or vertical line filtering, hole filling, object shadowing and others. An analytical method, based on the comparison principle for differential equations, has been proposed for synthesizing CNNs for simple transformations on bipolar images (Chua & Thiran, 1991). Nevertheless, when the desired transformation cannot be coded into a set of simple rules and greyscale pixels are used, GAs may represent a valuable alternative and overcome the typical limitations of methods based on gradient descent as in Chua et al. (1992).

3. GENETIC ALGORITHMS

GAs are stochastic similarity-based sampling techniques, especially suited for optimization problems in which little a priori knowledge is available about the function to be optimized. Often they have proved suitable for complex optimization problems, like combinatorial optimization, in which an analytical solution is not directly available or in which numerical techniques are misled by local minima. Their theoretical foundations lie simply in Darwin's evolutionary explanation of the genesis of species, i.e. they use simple concepts like the survival of the fittest or the genetic diversity in a given population as source of further species improvement. GA optimization has often been referred to as guided blind search; guided since a reinforcement signal drives it, but blind since it does not access the inside of the signal production itself.

Schematically, it works as follows: a coding is chosen to map any possible candidate solution of a given problem into a finite-size string (the *chromosome*) taken from some alphabet. An initial pool of such strings is randomly initialized and each of them is in turn evaluated, ranked according to its capability to solve the given problem. The latter is normally referred to as the *fitness* of the *individual*, and measures what in nature would be the individual's skills in positively interacting with the surrounding environment. The fitness ranking is then used for cloning the genetic material present in the population, i.e. the higher the fitness, the higher the chances that the individual gets its chromosome duplicated and used for mating with other individuals. Mating can be implemented in a variety of ways, but the basic mechanisms are the exchange of substring in the chromosome (*crossover*) and, with a low probability, a *mutation* of the same. The newborn individuals then replace totally or partially the old ones in the population, thus a new generation is built. This iterative process is stopped when the maximum fitness in the population does not increase further or has reached a satisfactory value. In either case, the best individual is taken as the solution. The choice of the coding strategy may be critical for the efficiency of the method but this issue, often questioned in the literature, is beyond the scope of this introduction. For a complete description see Goldberg (1989) and Davis (1991).

4. OUR GENETIC ALGORITHM

A real coded Genetic Algorithm has been used to guarantee a faster convergence and to allow a reduced-size population. Also, experience has shown how real number representation is more flexible when the range for the optimization parameters is not known in advance, as may be the case with some image processing tasks.

The following features have been used to enforce a more efficient reproduction phase.

- The best individual from the previous generation substitutes the worst in the current generation if no improvement is made (*elitism*).
- Fitness values are scaled linearly so as to stress small differences between similiar individuals. Scaling coefficients are computed every generation and depend upon the maximum, minimum and average of the current fitness. It is not uncommon that two individuals perform rather differently although their fitnesses are close.
- An individual gets exactly the number of offsprings he deserves in proportion to his rounded fitness (expected value model). This is to overcome stochastic errors connected with populations of small size and roulette-wheel selection schemes.

The current implementation of the GA does not systematically apply crossover and mutation to all the offspring. Instead, for each couple of new individuals one operator is selected to act on them according to its usage probability (actually percentage), as in Montana and Davis (1987).

Several of these operators have been tested. Mutation, for example, may occur basically in two ways, either totally replacing the specific gene with a new one randomly chosen from a given distribution, or simply adding a small perturbation to the pre-existing value (*creeping*). The latter is especially useful for performing stochastic hill-climbing in the parameter space and has often been quoted as the major advantage of the real coding strategy.

Crossover operators typically swap the corresponding genes in the parents, but they may also randomly choose one of the two to be put in both offspring. Other possibilities are linear combinations of the two values, for instance the average. *Couple* operators with probability 100% which always call both the desired crossover and mutation operators may be introduced. In addition a parameter is associated with each operator representing the probability of being applied to a single gene, once that operator has been chosen for usage. A mechanism to increase or decrease operators' usage probabilities when the convergence stalls has been implemented in order to keep the population diversity high in the later generations. Number of generations, population size and specific problem dependent parameters may also be adjusted.

5. PUTTING IT ALL TOGETHER

Supervised learning normally consists of presenting to the network a sequence of training patterns, namely couples of input and target patterns used to compute the average performance error. When the latter has become small enough the training phase is completed and the network may be used for operation. We have used in our simulations only one training couple since we expected the optimized CNN to be able to reproduce its performance on images sharing a similar greyscale distribution.

As in common GA practice, a population of P_s individuals is randomly initialized. An individual of such a population consists of A_{kl}, B_{kl}, C, R, I parameters, randomly chosen in the uniform intervals

$$A_{kl}, B_{kl}, I \in \left[-\frac{C_0}{2}, \frac{C_0}{2}\right] \text{ and } C, R \in [0, C_0].$$

Three kinds of symmetry for the template may be used: no symmetry, point symmetry and isotropy, reducing the number of genes per chromosome to $(2r+1)^2$, $r(2r+1)+r+1$ and $r+1$, respectively.

In each generation, all individuals are evaluated, i.e. its corresponding templates and parameters are used for solving (1) for T steps, namely until the system has reached the steady state (in this case the individual is considered stable) or the threshold of maximum dynamic steps has been exceeded (unstable individual). As a criterion, for the steady state, the settling down of the global magnetization

$$m(t) = \frac{1}{N} \sum_{(i,j)=(0,0)}^{I_W, I_H} y_{ij}(t), \tag{5}$$

has been used.

The simplest approach is to enforce **supervised learning** through the following fitness function for an individual x

$$f(w) = 1 - \frac{1}{N} \sum_{(i,j)=(0,0)}^{I_W, I_H} |y_{ij}(T, w) - r_{ij}| \tag{6}$$

where we have stressed the dependence of the steady state output value $y_{ij}(T,w)$ from the individual w. r_{ij} are the desired pixel greyscale values, I_W and I_H, respectively, the image width and height and $N = I_W \times I_H$.

Several other features may be taken into account, depending upon the specific task. Information entropy, magnetization, average correlation or other macroscopic features may be used for comparing the output pattern with the target, in pattern-matching applications as well as in ordinary image processing contexts. The variance of the above error may be used as well.

As long as each of these **fitness components** can be bounded, a more useful and general expression for the overall fitness of an individual w is therefore:

$$f(w) = \sum_{i=1}^{P} c_i f_i(w) \text{ with } \sum_{i=1}^{P} c_i = 1 \tag{7}$$

where P is the number of fitness components and the weights c_i have to be chosen by tentatives.

6. PARALLELIZATION

In this section, the double parallelism implemented on the Paragon XP/S 10 architecture is briefly explained.

A parallel GA can be implemented very easily by chosing P_1 different processors among the N_n available, to oversee P_l independent subpopulations, each with the same number of individuals (Tanese, 1987). These processors, called *masters*, perform the genetic operations on the genomes of the members of their subpopulations, each carrying out its own optimization procedure. If necessary, according to the task to be optimized, different initialization parameters can be chosen for each subpopulation, to guarantee that the initial individuals are localized in different regions of the search space and different genetic operators can be used by each master, if needed.

Similarly, the locality of the system specified in eqn (1), i.e. the fact that each $\dot{x}_{ij}$ is a function of its geometrical neighbours only, can be conveniently exploited to split the processing of the image into a certain number of independent slices. Suppose P_2 processors are chosen for

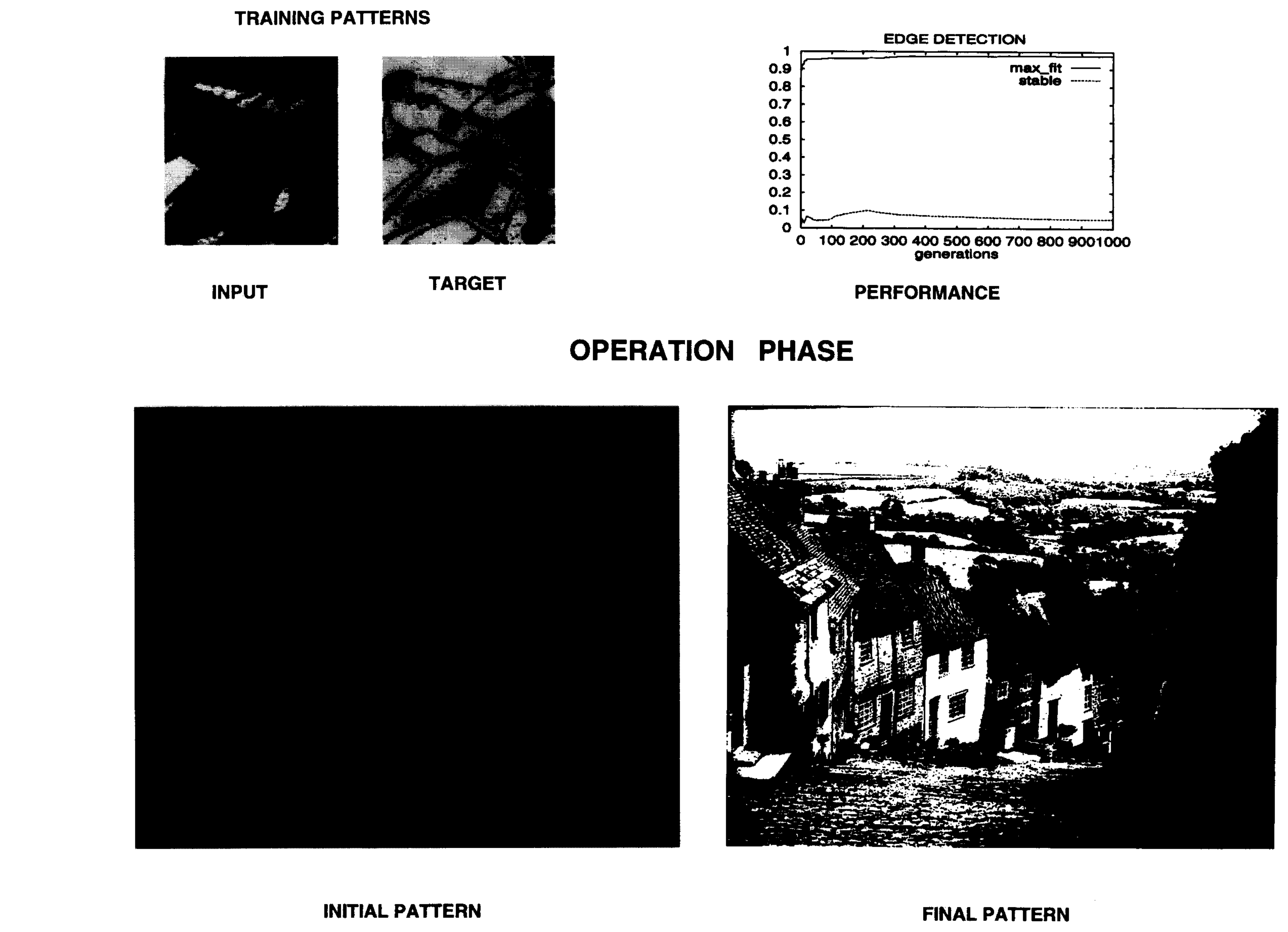


FIGURE 1. Training and operational phase for edge detection. The graph in the right top corner shows the behaviour of the GA optimization, plotting the fitness of the best individual in the population against the generation number. $P_S = 8$, $T = 10$, $C_0 = 2$, 1000 generations.

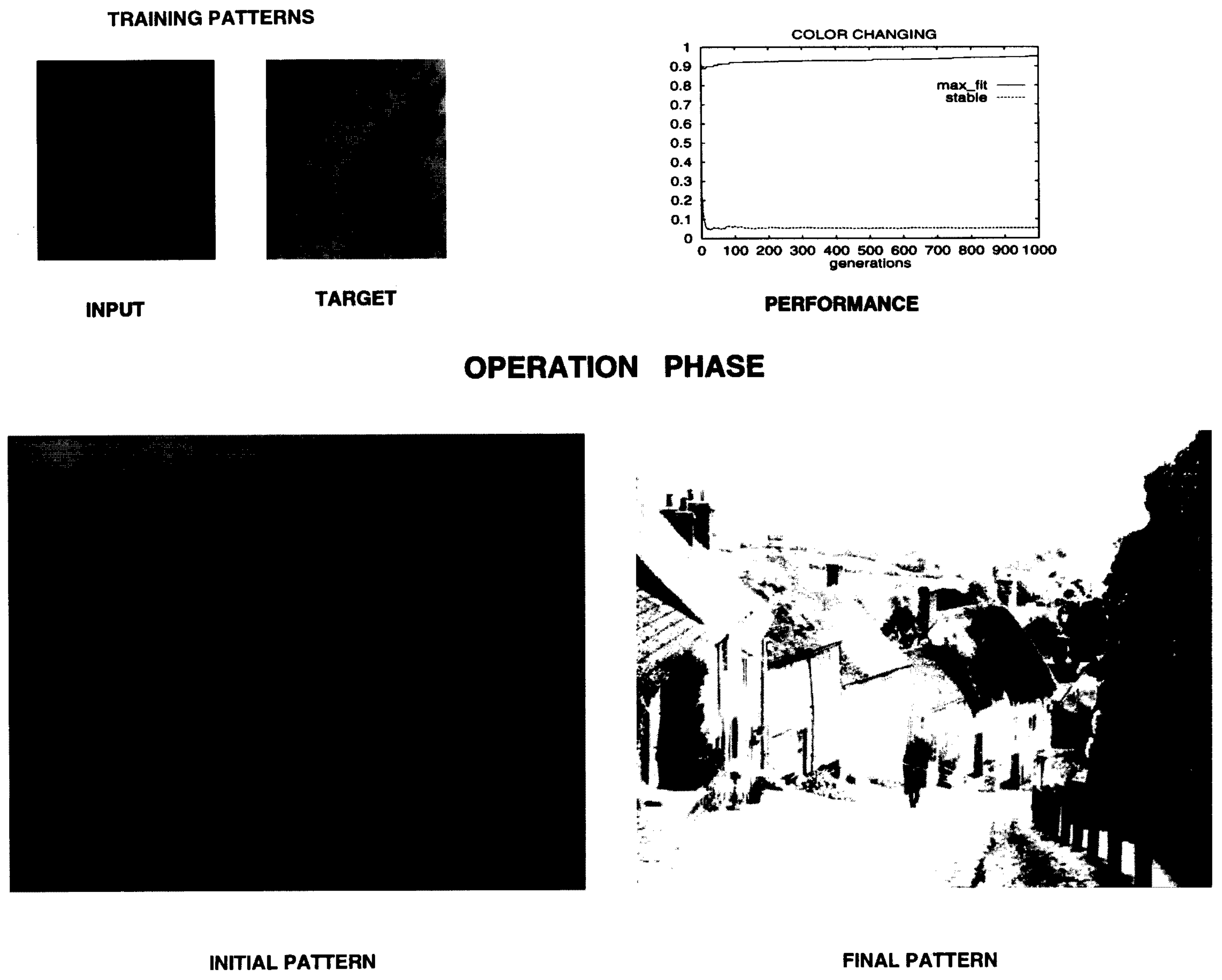


FIGURE 2. Training and operational phase for colour changing. The graph in the right top corner shows the behaviour of the GA optimization, plotting the fitness of the best individual in the population against the generation number. $P_S = 8$, $T = 10$, $C_0 = 4$, 1000 generations.

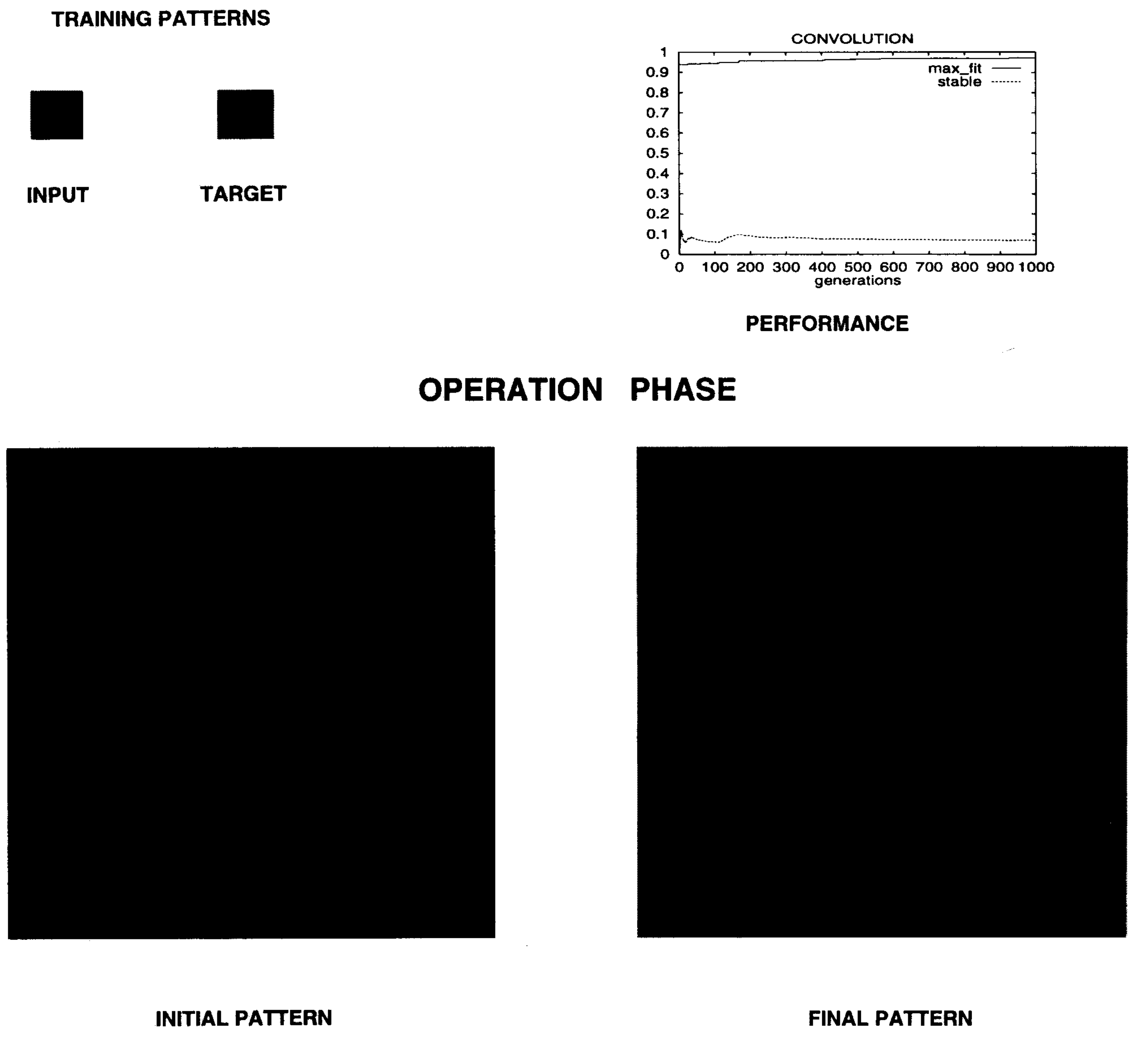


FIGURE 3. Training and operational phase for convolution with Mexican Hat operator. The graph in the right top corner shows the behaviour of the GA optimization, plotting the fitness of the best individual in the population against the generation number. $P_S = 8$, $T = 10$, $C_0 = 1$, 1000 generations.

this task, each being assigned to a single slice of the image. The image can be divided into stripes so that each processor oversees a rectangular area and has, at most, a left and a right neighbour. For each integration step, each processor oversees the dynamic (1) in its own slice only and merely needs to exchange the information contained on the borders with its own neighbours. The width of this border is equal to the radius r. The integration is carried out for the desired maximum number of steps or until the criterion for the steady state is fulfilled. In this way, the dynamic in eqn (1) and thus the computation of the fitness (6) for a single individual is parallalized with P_2 different processors, henceforth called *slaves*.

A number P_2 of unique slaves is assigned to each master for the parallel evaluation of the individuals in its subpopulation. Within each master, members of the subpopulations are evaluated sequentially, one after the other, but each evaluation takes place in parallel.

In order to prevent the master from idling in a wait state during each evaluation, its role is changed into slave so that it can actively contribute to the computation (in this sense master processors are in fact master and slave). The number of masters and slaves thus fulfils $N_n = P_1 \times P_2$. After each individual in the subpopulation has been ranked according to its fitness, reproduction, crossover and mutation take place through the usage of the different genetic operators until the next generation is built up.

All communication within the P_1 subgroups takes place via the master nodes which are disposed on a ring topology so that each one has a left and right neighbour. With a desired frequency they exchange their best individuals by passing them rightwards along the ring. This prevents too much computational power from being lost by one group of nodes when another master is already optimizing in a more promising region of the search space (Tanese, 1987). This technique may be convenient for best exploiting computing power when the number of processors handling the image has already exceeded the speed up threshold and increasing the number of slave processors would make it no better.

During the initialization phase, all processors are assigned to their specific roles (master or simply slave) and grouped together in P_1 subgroups (each with one master/slave and $P_1 - 1$ slaves). The whole image is then subdivided and transmitted from node zero, which accesses mass storage devices, to all the others and the computation can start.

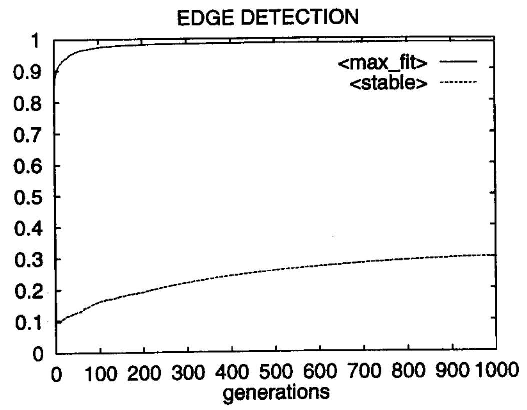
7. SIMULATION RESULTS

We applied the system described in Section 5 to construct templates implementing an edge detection filter, a colour palette changing filter and one reproducing the convolution between the image and the second derivative of a Gaussian (the so-called Mexican hat operator, used in MRI images for edge detection) (Ehricke, 1990).

The maximum theoretical value for fitness (6) is 1, meaning that the desired mapping was learnt error-free. Although such a value was never reached in our runs, results are satisfactory and the method proved to be valid, at least for the desired tasks. Its efficacy may be easily inspected from Figures 1–3, showing examples of training and operation phases as well as graphs of maximum fitness in population against generation number.

The use of analytical methods for optimizing CNN connectivity in pattern recognition tasks has been so far limited to bipolar images, preventing therefore a direct comparison on the quality of obtained results, and the study of convergence speed of numerical methods versus genetic ones was not in the scope of this work.

In all the experiments a population of eight individuals was evolved for 1000 generations. The neighbourhood radius r was set to 1 in all cases. The maximum fitness function plots in Figure 4 behave consistently



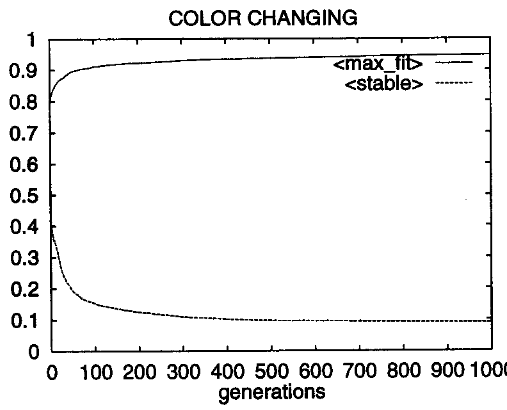


FIGURE 4. Averaged maximum fitness and stabilized percentage (rescaled to unity) for edge detection and colour changing tasks. Parameters are: $T = 6$, $P_S = 8$, $C_0 = 1$, 1000 generations.

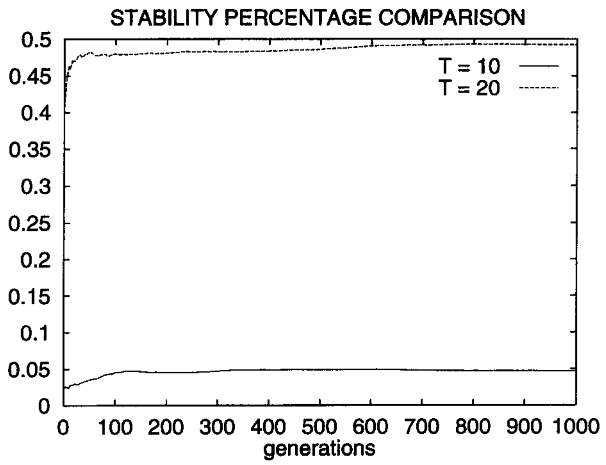


FIGURE 5. Stabilized percentage behaviour for different maximum number of steps for convolution task.

```
# S1 1 S2 3
C: 3.400787 R: 0.168945 I: -0.270355

   1.372711   1.952759    0.340668
   0.821655  -1.285492    0.744843
   1.012238  -1.410400   -1.085815

   0.959442  -0.422333   -1.316528
  -1.274567  -1.192474    0.162292
  -0.318390  -0.578308   -0.223236


# FITNESS 0.896841
```

FIGURE 6. An example of CNN templates for the convolution filter with $r = 1$.

with all GA results to date, namely, after initial rapid improvements, the diversity in the population is reduced and the convergence stalls.

As for other real-coded GAs, mutation played a major role in our system and was granted 60% of the usage percentage. In particular the operator adding a small perturbation to the genes and the one totally replacing them with a certain probability have shown the best performance for all the applications. The remaining 40% of usage percentage was given to a crossover operator that swaps the corresponding genes in the parent chromosomes.

The maximum number of time steps T plays an important role, as can be seen from Figure 4 which shows how genetic pressure can lead to stable or unstable individuals according to the different tasks. The effects of increasing T on the percentage of stable individuals can be seen for the convolution task in Figure 5.

The templates obtained after 1000 generations (see for instance Figure 6) were tested in the operational phase and worked equally well on the training pattern as on several other images of the same kind.

The training patterns used were approximately 1000 × 1000 points large, later reduced by one order of magnitude to decrease evaluation times. Nevertheless some care must be taken to choose a sample containing all the necessary features so that the network can learn the desired mapping, for example in pattern classification tasks.

The fitness used in the previous examples can be quite small even when results are rather different from expected, as can be seen in Figure 7.

8. CONCLUSIONS AND FUTURE WORK

Besides the simplicity of the transformations shown in this paper, we have shown that GAs can be used successfully for CNN training even with relatively small target patterns which need to be constructed using computationally more expensive techniques.

Unfortunately, real-coded GAs can be hindered from reaching the best fit individual if the problem is *deceptive* (see Goldberg, 1991), however, using the slower but more robust *messy* GAs (Goldberg et al., 1989) might provide a method for determining whether or not the CNN can perform a given task.

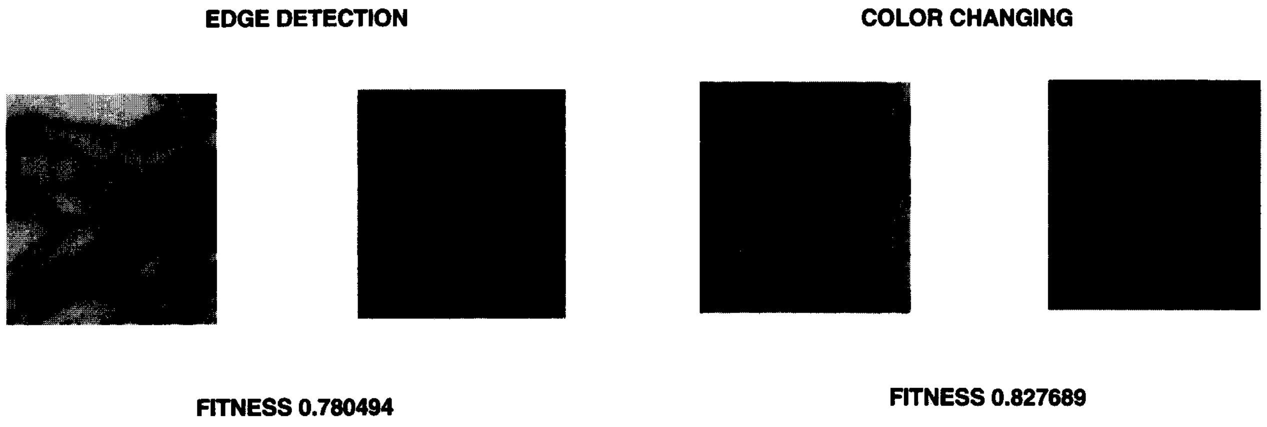


FIGURE 7. Examples of individual performance at generation zero.

REFERENCES

Bornholdt, S., & Graudenz, D. (1992). General asymmetric neural networks and structure design by Genetic Algorithms. *Neural Networks*, **5**, 327–334.

Chua, L. O., Schuler, A. J., Nachbaer, P., & Nossek, J. A. (1992). Learning state space trajectories in cellular neural networks. *Proceedings of the Second IEEE International Workshop on Cellular Neural Networks and their Applications*, IEEE 92TH0498-6.

Chua, L. O., & Thiran, P. (1991). An analytic method for designing simple cellular neural networks. *IEEE Transactions on Circuits and Systems* **38**, 132–1341.

Chua, L. O., & Yang, L. (1988). Cellular Neural Networks: theory. *IEEE Transactions on Circuits and Systems*, **35**, 1257–1272.

Davis, L. (1991). *Handbook of genetic algorithms*. New York: Van Nostrand Reinhold.

Ehricke, H.-H. (1990). Problems and aproaches for tissue segmentation in 3D-MR imaging. SPIE Vol. 1233 *Medical Imaging IV: Image Processing*.

Goldberg, D. E. (1989). *Genetic Algorithms in search, optimization and machine learning*. Reading, MA: Addison-Wesley.

Goldberg, D. E. (1991). Real-coded Genetic Algorithms, virtual alphabets and blocking. *Complex Systems*, **5**, 139–167.

Goldberg, D. E., Korb, B., & Deb, K. (1989). Messy Genetic Algorithms: motivation, analysis and first results. *Complex Systems*, **3**, 493–530.

Montana, D. J., & Davis, L. (1987). Training feedforward neural networks using genetic algorithms. *Proceedings of the 1989 International Joint Conference on Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann Publishers.

Tanese, R. (1987). Parallel Genetic Algorithm for a hypercube. *Proceedings of the 2nd International Conference on Genetic Algorithms*. Lawrence Erlbaum Publishers.