Automatic Design of Cellular Neural Networks by means of Genetic Algorithms: Finding a Feature Detector

Frank Dellaert [1] and Joos Vandewalle [2]

[1] Dept. of Computer Engineering and Science
Case Western Reserve University
10900 Euclid Avenue, Cleveland, OH 44106
e-mail: dellaert@alpha.ces.cwru.edu

[2] Dept. of Electrical Engineering
Katholieke Universiteit Leuven
Kardinaal Mercierlaan 94, 3001 Heverlee, Belgium
vandewalle@esat.kuleuven.ac.be

Abstract - This paper aims to examine the use of genetic algorithms to optimize sub-systems of cellular neural network architectures. The application at hand is character recognition: the aim is to evolve an optimal feature detector in order to aid a conventional classifier network to generalize across different fonts. To this end, a performance function and a genetic encoding for a feature detector are presented. An experiment is described where an optimal feature detector is indeed found by the genetic algorithm.

1 . Introduction

We are interested in the application of cellular neural networks in computer vision. Genetic algorithms (GA's) [1-3] can serve to optimize the design of cellular neural networks. Although the design of the global architecture of the system could still be done by human insight, we propose that specific sub-modules of the system are best optimized using one or other optimization method. GA's are a good candidate to fulfill this optimization role, as they are well suited to problems where the objective function is a complex function of many parameters.

GA's have been used before in the design of neural networks, e.g. [4-7] and even specifically related to vision: for example [8,9] where the emphasis is on vision-based behavior.

The specific problem we want to investigate is one of character recognition. More specifically, we would like to use the GA to find optimal feature detectors to be used in the recognition of digits .

2. Problem

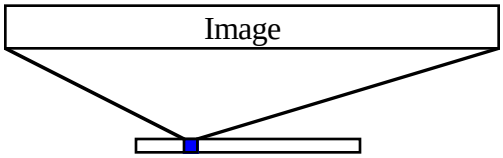

Figure 1. Training a neural network directly on the image with a simple delta rule does not yield a generalizing net.

Suppose we want a neural network that can correctly recognize the digitized characters 0...9 regardless of their font: one way to do this is to wire 10 neurons to all pixels of the image and train the network with a simple delta rule to try and minimize the system error (Fig.1). Of course this will work, but the delta-rule will quickly take advantage of the specific properties of the training data and will not be able to generalize recognition to digits of another font. In addition it is questionable to work directly from a pixel image.

A better approach would be to pre-process the image to extract features from it, e.g. 'end of line' or 'small blob', and train the network to recognize the position of these features: that way it can associate those features with a specific digit, rather than the pixels. The idea is of course that while specific pixels can vary between digits of different fonts, the approximate position of the features will remain the same and thus can be used to train the network. Even using the very simple delta rule, one would expect a better generalization of such a network.

In this paper, we will be interested in finding the best feature detectors, i.e. those that result in networks that can best generalize to data *other than the training data*. To that end, we will use the genetic algorithm, but first we need to define what a 'feature detector' is and how we can test whether it is a 'good' one.

3. Evaluation of a Feature Detector

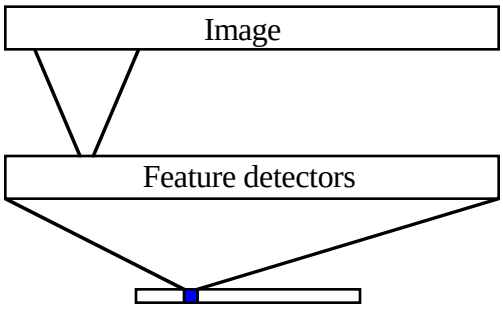

Figure 2. The simplest architecture where the image is pre-processed by one feature detector.

To aid the classifier network in correctly recognizing digits, we will interpose a number of pre-processing layers between the image layer and the classifier network. We will call these layers *feature detectors*, as high activity in a neuron of such a layer would correspond to the existence of a feature at the same location in the image layer.

The simplest instance of this arrangement just uses one feature detector, as depicted in Fig.2. Both the image and feature-layer are neuron-lattices of 11 by 8, whereas the classifier-layer consists of 10 neurons, one for each digit. The neurons in the feature layer are all connected to a 5 by 5 receptive field in the image layer using a fixed weight template (the same for all neurons in the layer). This template completely defines the feature detector, and we will use the GA to find the

optimal weight template. All the neurons simply add their input and apply a sigmoidal transfer function to calculate their output:

$$y = g\left(\sum_j w_j o_j - \theta\right), \text{ where } g(x) = (1+e^{-x})^{-1}$$

When given a feature detector, we can calculate a *performance value* for it, indicating how good (or bad) it is at generalizing over different fonts. This happens in two steps:

(1) The classifier-layer is trained to recognize the digits in the *training data* using the simple delta-rule (3 presentations of each digit):

$$\delta = \delta_k \cdot y(1-y)\cdot(y-d),\ \Delta w_j = -\delta \cdot o_j \text{ and } \Delta\theta = \delta$$

where y is the output of a classifier neuron, d the desired output, o_j the output of the feature neurons and δ_k the learning speed. The training data is shown in Fig.3.

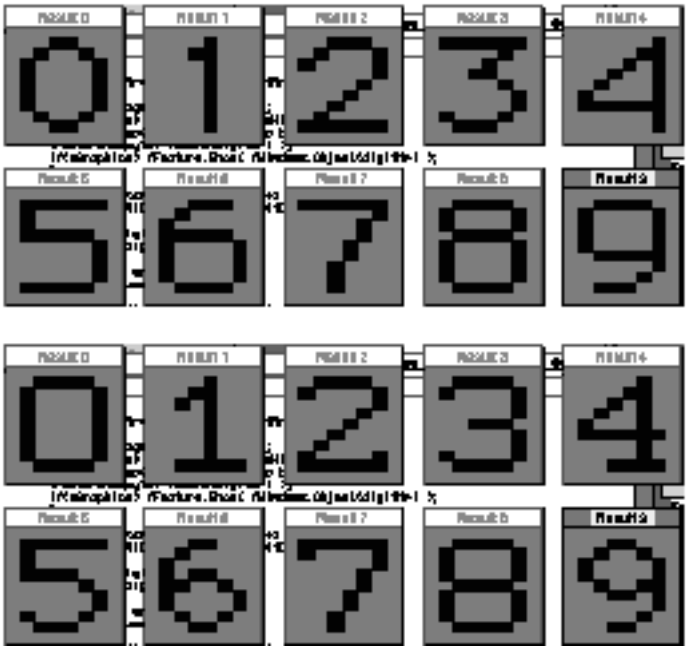

Figure 3. Top: the training data, based on a 12 pt Geneva. Bottom: the control data, Courier.

(2) The network is evaluated using the *control data* without any additional training, and this will give us the performance value for the feature detector, calculated as follows:

$$P = \sum_{d=0}^{9} p_d \qquad \begin{cases} p_d = 10 + s_d, \max(y) \neq y_d \\ p_d = 0, \max(y) \neq y_d \end{cases}$$

where y_j is the output of classifier neuron j when digit d is presented, and s_d is the selectivity of the network for digit d, a measure for how strong the network differentiates between the correct digit and the other digits:

$$s_d = y_d - \tfrac{1}{9}\sum_{j\neq d} y_j$$

Note that the total performance of the network consists of two components: a multiple of 10 that will indicate how many digits were correctly recognized, and a smaller component that represents the average selectivity for the recognized digits.

4. The Genetic Algorithm

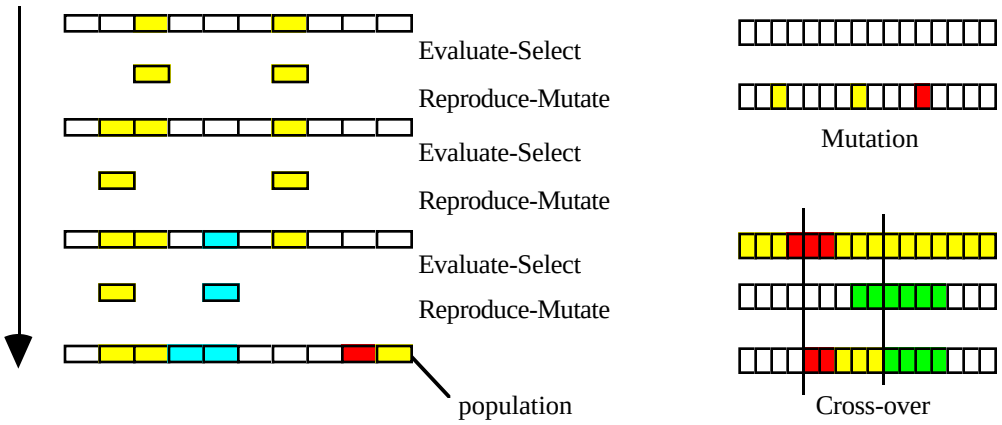

Figure 4. The sequence of events in the GA used.

Genetic algorithms are optimization algorithms that work according to a scheme analogous to that of natural selection. For the GA to be applicable, it is assumed that any valid feature detector can be described by a vector of real numbers, hereafter called the *genome*. The GA works with a *population* of such genomes, and alternates evaluation, selection, reproduction and mutation over the population to search the space of possible feature detectors. Any genetic algorithm can be summarized as follows (see Fig.4 for a graphical illustration of the process):

1. Generate a random population of genomes
2. *Evaluate* the performance of each genome by testing the performance of its associated feature detector
3. *Select* the genomes with the best performance values
4. *Reproduce* using the selected genomes to replenish the population, at the same time producing variation by mutation and/or cross-over
5. Go to step 2 or exit when done

The GA we use is different from the 'classical' GA approach in the sense that reproduction is localized: all the genomes are arranged along one dimension, as shown in Fig.4, and after selection the genomes that will become parents retain their position. Mutated offspring is generated only in the interval dominated by the parent, and likewise cross-over happens only between two adjacent parents. We have found this arrangement to be less prone to premature convergence (one genome taking over the entire population) than other schemes.

Mutation is implemented by adding a random unit vector to the genome, with its length chosen from a Gaussian distribution with a specific variance (in our case 0.2). Cross-over is done between two parents by exchanging a part of the genome-vector with another genome-vector (two-point cross-over). This was done with a probability of 20%.

Thus, to optimize feature detectors we still lack one crucial element, i.e. a way to encode a feature detector in a genome. This will be discussed in the next section.

5. Encoding of a Feature Detector

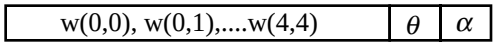

Figure 5. The encoding of the feature detector on the genome.

We can describe the wiring from the image layer to the feature layer simply by supplying the weight matrix and the threshold used. This gives us a natural encoding for the feature detector in a genome: we simply line up the 25 weights and the threshold and we have a vector of real numbers. A scaling factor is also added (so that the overall level of connection strength can easily be found by the GA), giving us the straightforward genetic encoding for a feature detector as shown in Fig.5.

6. Results

Using the approach described in the previous sections we have indeed obtained a feature detector that could recognize all 10 digits of the control set. This is a definite improvement with respect to the network without the feature detector (i.e. delta rule on wiring from image to classifier directly): although that network recognized all digits in the training data, it could only identify three digits in the control set.

The evolved detector was the best of its generation when we stopped the GA after 1000 generations, and its performance value was 104.1, i.e. with an average selectivity of 0.41. The performance value when tested with the training data was higher, 106.8, to be expected as the network was trained with that data.

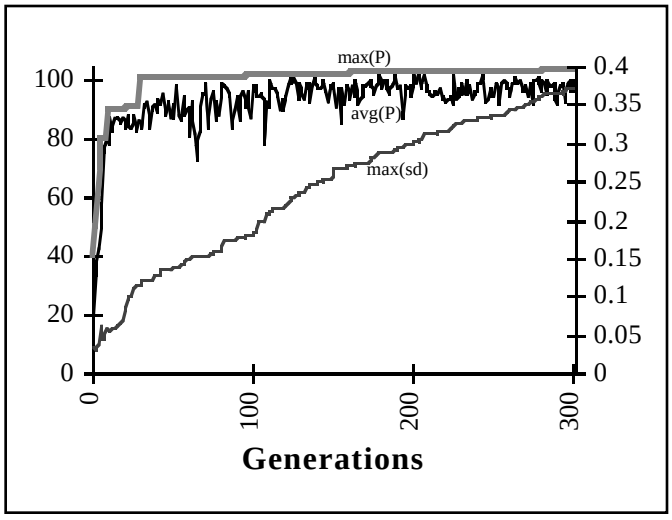

Figure 6. The avg. performance (black line), max. performance (gray) and max. selectivity (slowly rising) of each generation during the run of the GA.

By looking at the first 300 generations of the evolutionary process (Fig.6) we can see that a detector recognizing all digits correctly in the control data is discovered quite rapidly. The average selectivity rises steadily throughout the process, and interestingly, it is conserved when the detector manages to recognize one digit more: there is no dip in the average selectivity to indicate that the recognition of an extra digit extracts a performance penalty on the selectivity of the network.

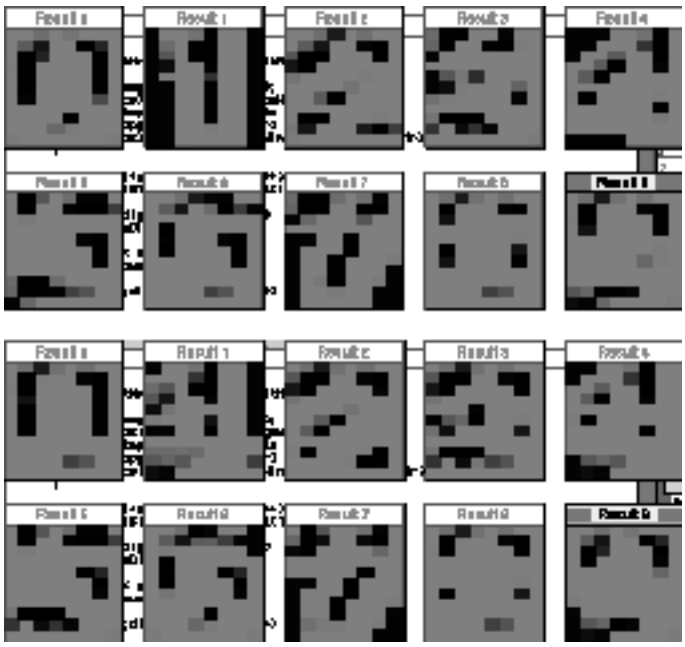

Figure 7. The evolved feature detector applied to training data (top) and control data (bottom). Gray corresponds to an activity of 0, white to -1 or less, and black to +1 or less.

In Fig.7 the output of the feature neurons is shown when they are wired to the image using the evolved weight template. Note again that the classifier is trained using these output values for the training data, but the performance is determined with those of the control set. Thus, the evolved feature detector does its job by discovering features that are common to both fonts.

7. Discussion

We have shown that GA's can be used successfully in cellular neural network design, i.e. we can evolve the detailed weight matrix of one cell in the global, human designed architecture. This might enable us to improve the performance of neural network solutions in specific applications at a reduced cost, since the role of expensive human design is decreased.

However, the work described here is an early exploration in this area, and is open to some criticisms. Thus, in future work we would like to confirm these results by using larger data sets, divided in three categories: training data, evaluation data to be used in the longer term learning process (evolution) and control data, to test whether the feature detector evolved really finds universal features of characters, not only the common features between the first two sets.

We also think that it would be worthwhile to extend this work towards the evolution of multiple feature detectors and pre-processing layers: we feel that a classifier network will make maximum use of combinations of features in an image and that this will boost the performance of the overall net towards larger data sets (e.g. all ASCII characters).

Finally, though in the experiment described a classifier network was used as the last recognition step, we suspect that the ability of the GA to cope with complex performance functions will enable us to easily substitute other methods here, e.g. statistical methods as they are often used in commercial OCR packages. In conclusion, we think that automatic design by means of GA's shows considerable promise as an engineering tool with wide applicability.

References

[1] J. H. Holland, “Adaptation in Natural and Artificial Systems”, University of Michigan Press, Ann Arbor, MI, 1975

[2] D. E. Goldberg, “Genetic Algorithms in Search, Optimization and Machine Learning”, Addison-Wesley, Reading, Mass., 1989

[3] T. Kozek, T. Roska and L. O. Chua, “Genetic Algorithm for CNN Template Learning”, *IEEE Trans CAS I*, vol. 40, pp. .392-402, 1993

[4] R. D. Beer and J. C. Gallagher, “Evolving dynamical neural networks for adaptive behavior”, *Adaptive Behavior*, vol. 1, pp. 91-122, 1992

[5] M. A. Lewis, A. H. Fagg and A. Solidium, “Genetic Programming Approach to the Construction of a Neural Network for Control of a Walking Robot”, in *IEEE International Conference on Robotics and Automation in Nice, France*, pp. 2618-23, 1992

[6] G. F. Miller, P. M. Todd and S. U. Hegde, “Designing neural networks using genetic algorithms”, in *Proceedings of the Third International Conference on Genetic Algorithms in George Mason University*, San Mateo, CA, pp. 379-384, 1989

[7] R. Belew, J. McInerney and N. N. Schraudolph, “Evolving networks: Using the genetic algorithm with connectionist learning”, University of California, San Diego, CSE Technical Report CS90-174, 1990

[8] I. Harvey, P. Husbands and D. Cliff, “Seeing The Light: Artificial Evolution, Real Vision”, in *From Animals to Animats 3, Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, Dave Cliff, Philip Husbands, Jean-Arcady Meyer, and Stewart W. Wilson (ed.), pp. 392-401, MIT Press, Cambridge, MA, 1994

[9] C. Reynolds, “An evolved, vision-based model of obstacle avoidance behavior”, in *Artificial Life III*, Christopher G. Langton (ed.), pp. Addison-Wesley, Reading, MA, 1994