Staged Training of Neocognitron by Evolutionary Algorithms

Zhengjun Pan[1,2] **Theo Sabisch**[1] **Rod Adams**[1] **Hamid Bolouri**[1]

[1]Science & Technology Research Centre, and
Department of Computer Science
University of Hertfordshire, Hatfield, Herts, AL10 9AB, UK
{Z.Pan, T.Sabisch, R.G.Adams, H.Bolouri}@herts.ac.uk

[2]State Key Laboratory of Software Engineering
Wuhan University, Wuhan, Hubei, 430072, P.R.China

Abstract- The Neocognitron, inspired by the mammalian visual system, is a complex neural network with numerous parameters and weights which should be trained in order to utilise it for pattern recognition. However, it is not easy to optimise these parameters and weights by gradient decent algorithms. In this paper, we present a staged training approach using evolutionary algorithms. The experiments demonstrate that evolutionary algorithms can successfully train the Neocognitron to perform image recognition on real world problems.

Key words: Pattern recognition, Neocognitron, neural network, medical image, learning, evolutionary computation

1 Introduction

A key issue for visual pattern recognition is that the objects to be recognised are generally subjected to various forms of transformation. Thus, development of an artificial visual system with the ability to tolerate variations in position, scale, shift and rotation has motivated researchers to propose a number of methods. These approaches to shift and deformation tolerance fall into three categories, learning the transformations, extracting invariant features and building a specific architecture. Roughly speaking, the first approach is to learn the object models from examples [Hinton, 1987] and try to match the observed and stored models by determination of the transformation parameters [Chin and Dyer, 1986]. For example, each object will be presented at many different positions if translational invariance is desired. Typically, such methods require a carefully prepared, and very large data set. The second approach is to use feature spaces which are automatically invariant to some transformations such as moments [Abutaleb et al., 1989] and polar-coordinate Fourier transform [Shridhar and Badreldin, 1984]. Because feature spaces are known to be problem dependent, the feature extraction stage forms an integral part in the recognition system and all images need to undergo the filtering process. Depending on the number and type of the required invariant transformations, the processing power required can be prohibitively large.

In the third approach, transformation invariance is embedded in the structure of a distributed processor network for parallel processing. This approach was inspired by the physiology of biological visual systems. The Neocognitron of Fukushima [Fukushima, 1988] and the zip code recognition system of LeCun [LeCun and Farber, 1989] are representative examples. They use hierarchical networks which contain several successive layers. The inputs to the network are given by the intensities of the pixels in a two-dimensional array. Units in each layers are similarly arranged in two-dimensional sheets to reflect the geometrical structure of the problem. Instead of having full interconnections between adjacent layers, each unit receives inputs only from units in a small region in the previous layer, known as a receptive field. The use of receptive fields can dramatically reduce the number of weights present in the next work compared with a fully connected architecture. This makes it practical to treat pixel values in an image directly as inputs to a network.

For the last 2 decades, the Neocognitron has been acclaimed as a shift and distortion tolerant character recognition system. However, not much independent empirical research has been published on the use of this system for real world problems [Lovell, 1994, Lovell et al., 1997, Sabisch, 1998]. The main reason appears to be that the architecture is too complicated to train and its performance is too sensitive to various parameters. Thus, training robustness has hindered the application of the Neocognitron to real world problems. Fukushima himself trained the Neocognitron by either supervised or unsupervised learning schemes (Fukushima called them learning with a teacher and learning without a teacher respectively.). For unsupervised learning, several typical examples are presented to the network without providing any information as to which category they belong to, while in supervised learning, the teacher clusters similar input features to each detector type. Many researchers have claimed that successful training of Neocognitron required judicious choice of network parameters such as feature selectivity and learning rate [Lovell, 1994]. Even the or-

0-7803-5536-9/99/$10.00

der in which example patterns are presented to the network affects the training results. Also Fukushima's supervised and unsupervised learning algorithms are competitive learning and can rarely find the global optimum.

Teo *et al* in [Teo et al., 1997] used a Genetic Algorithm to globally optimise Neocognitron network parameters and reported promising results. However, their approach required the manual adjustment of several new parameters introduced in the process. Thus, their approach did not resolve the issue of manual parameter selection.

In this paper, we use evolutionary algorithms for numerical optimisation [Pan and Kang, 1997] to train the weights and training parameters of the Neocognitron. In sections 2 and 3, we will describe the architecture of the Neocognitron and our staged training algorithm respectively. In subsequent sections, we will present our test problem and experimental results and conclude the paper.

2 Network architecture of the Neocognitron

The Neocognitron [Fukushima, 1988, Lovell, 1994] is an example of a hierarchical neural network in which there are many layers, with a very sparse and localised pattern of connectivity between the layers. It was designed to recognise handwritten characters — specially, the Arabic numerals $0, 1, \cdots, 9$. The purpose of the network is to make its response insensitive to variations in the position and style in which the digits are written.

The architecture of the Neocognitron consists of several pairs of layers as in figure 1. The first layer in each pair is an S-layer, the second layer is denoted a C-layer. Layers from the input to the output are referred to as $U_{C0}, U_{S1}, U_{C1}, U_{S2}, U_{C2}, U_{S3}, U_{C3}, U_{S4}, U_{C4}$ (assuming we only have four pairs of layers after the input layer). As we can note from figure 1, the input cells are arranged in a single rectangular array and cells within other layers are arranged in a number of rectangular arrays (called planes). The motivation for the multiple copies of the arrays in each S-layer is for each plane to respond a particular feature or group of features from the original input. Each S-cell in a particular plane tries to respond to that feature in a small portion of the previous layer, i.e., a receptive field. Then the C-cells combine the results from related S-planes and simultaneously thin out the number of cells in each array.

There are three different types of cells (or neurons) in Neocognitron — S and C cells, which are cells in S-layers and C-layers, respectively, and V cells (not shown in figure 1), which merely serve to normalise the activities of the S-cells. A cell in one layer receives signals from its receptive field in the previous layer, and sends signals to only a few cells in the next layer. In general, the size of the arrays decreases as we progress from the input layer

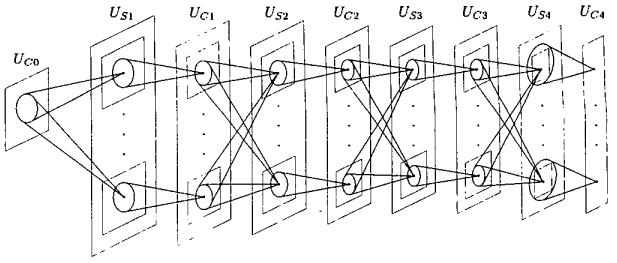


Figure 1: The architecture of a Neocognitron.

to the output layer of Neocognitron.

For each layer except input layer, the interconnection between this layer and its previous layer is determined by four sets of parameters: (1) number of planes, (2) dimensions of a plane, (3) size of the receptive field, and (4) the overlapping step size, i.e., number of neurons which are shared by two adjacent receptive field in each direction.

Following Fukushima's terminology [Fukushima, 1988, Lovell, 1994], weights from C to S-cells are denoted as $a_l(\boldsymbol{\nu}, \kappa, k)$, where l is the layer of the S-plane that the link connects to, k is the serial number of that S-plane, κ is the serial number of the C-plane from which the link originates and $\boldsymbol{\nu}$ is the location within the receptive field A_l of the C-cell from which the link originates.

An S-cell receives excitatory signals from its receptive field in the previous layer (a C-layer) and an inhibitory signal from a V-cell. The degree that V-cells affect cells in a given S-plane k is determined by the positive value of the inhibitory coefficient $b_l(k)$.

The output of an S-cell in the kth S-plane of the lth layer of the Neocognitron is given by

$$u_{Sl}(\boldsymbol{n}, k) = r_l \cdot \varphi\left[\frac{1 + \sum_{\kappa=1}^{K_{Cl-1}} \sum_{\boldsymbol{\nu} \in A_l} a_l(\boldsymbol{\nu}, \kappa, k) \cdot u_{Cl-1}(\boldsymbol{n} + \boldsymbol{\nu}, \kappa)}{1 + \frac{r_l}{r_l + 1} \cdot b_l(k) \cdot u_{Vl}(\boldsymbol{n})} - 1\right] \quad (1)$$

where r_l is the feature selectivity parameter for the l-th S-layer, which determines how closely the cell's input must correspond to the inputs it was trained on in order to elicit a response, and $\varphi(\cdot)$ is referred to as a threshold-linear transfer function and is defined by

$$\varphi(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 \leq x, \end{cases} \quad (2)$$

and u_{Vl} is the output of a V-cell and is defined by

$$u_{Vl}(\boldsymbol{n}) = \sqrt{\sum_{\kappa=1}^{K_{Cl-1}} \sum_{\boldsymbol{\nu} \in A_l} c_l(\boldsymbol{\nu}) \cdot u_{Cl-1}^2(\boldsymbol{n} + \boldsymbol{\nu}, \kappa)} \quad (3)$$

where $c_l(\boldsymbol{\nu})$ is the weight between a V-cell in layer l and the previous C-plane.

A C-cell's function is to allow for position variations in the features of the stimulus. Each C-cell receives signals from a group of preceding S-cells, which extract the same feature, but from slightly different positions. The C-cell is activated if at least one of these S-cells is active. Even if the stimulus feature is shifted in position and another S-cell is activated instead of the first one, the same C-cell keeps responding. Hence, the C-cell's response is less sensitive to shifts in the position of the input pattern.

The output of a C-cell on the receptive field D_l is defined by

$$u_{Cl}(\boldsymbol{n},k) = \psi\left[\sum_{\kappa=1}^{K_{Sl}} j(\kappa,k)\cdot \sum_{\boldsymbol{\nu}\in D_l} d_l(\boldsymbol{\nu})\cdot u_{Sl}(\boldsymbol{n}+\boldsymbol{\nu},\kappa)\right] \tag{4}$$

where

$$\psi(x) = \frac{\varphi(x)}{1+\varphi(x)} \tag{5}$$

and $j(\kappa,k) = 0$ or 1 depending on the connectivity between the κ-th C-plane and the k-th S-plane of the l-th layer. In our simulations, the number of S-planes is the same as the number of C-planes in each pair of layers and we will have $j(\kappa,k) = 1$, if $\kappa = k$, otherwise, $j(\kappa,k) = 0$.

We know from the above definitions that there are four different kinds of weight used in the Neocognitron: $a_l(\boldsymbol{\nu},\kappa,k), b_l(k), c_l(\boldsymbol{\nu}), d_l(\boldsymbol{\nu})$. The first two of these are determined during training and the last two are defined by

$$c_l(\boldsymbol{\nu}) = \gamma_l^{|\boldsymbol{\nu}|}, \tag{6}$$

$$d_l(\boldsymbol{\nu}) = \bar{\delta}_l\cdot\delta_l^{|\boldsymbol{\nu}|} \tag{7}$$

where $0 < \gamma_l, \delta_l \leq 1$ and $0 < \bar{\delta}_l$ define the roll-off of the receptive field weights away from the centre of the field. In the existing literature, the parameters $\gamma_l, \delta_l, \bar{\delta}_l$ are usually predefined, but since they affect the network's performance, ideally they should be determined during our training. For the mask distance $|\boldsymbol{\nu}|$, there are several implementations [Lovell, 1994]. Here we will use Fukushima's approach described in [Fukushima and Wake, 1991, Lovell, 1994]. That is, $|\boldsymbol{\nu}|$ is the Euclidean distance to the centre of the receptive field, and the C-cell mask is normalised as

$$\sum_{\kappa=1}^{K_{Sl}}\sum_{\boldsymbol{\nu}\in D_l} c_l(\boldsymbol{\nu}) = 1.$$

3 Trainable Network Parameters and Weights

In this paper, we assume the network architecture (i.e. the number of planes, the dimensions of each plane, the size of the receptive fields, and the degree of receptive field overlap) is preset by user-analysis of application characteristics. All other network parameters and weights will be adjusted automatically by our evolutionary procedure. Specifically, in each stage of the Neocognitron, the excitatory and inhibitory weights from C-cells to S-cells are variable and weights from S-cells to C-cells are controlled by the parameters $\delta_l, \bar{\delta}_l$. Other parameters to be determined include γ_l and the selectivity parameter r_l.

We will only discuss our implementation of supervised training. In this approach, the teacher (i.e. user) has to assign feature classes to each feature detector plane prior to training.

3.1 Staged training

To reduce the computational load, we train the Neocognitron stage by stage, as follows. First, we use the receptive field size of layer U_{S1} to cluster features in the input images into as many classes as there are detector planes in layer U_{S1}. Each U_{S1} plane is assigned to detect one group of features. With the features as the inputs and their responses as the targets, we find the parameters and the weights of layer U_{S1} by our evolutionary algorithm (described below).

Subsequent stages in the network are trained in a similar manner using the outputs of the preceding stage as the training input patterns. The target output patterns for the last stage represent the desired classification of the input images.

Suppose the network architecture consists of four pairs of processing layers, then we can describe our staged training procedure as:

$$U_{C0} \rightarrow U_{S1}$$

$$U_{C0} \rightarrow U_{S1} \rightarrow U_{C1} \rightarrow U_{S2}$$

$$U_{C0} \rightarrow U_{S1} \rightarrow U_{C1} \rightarrow U_{S2} \rightarrow U_{C2} \rightarrow U_{S3}$$

$$U_{C0} \rightarrow U_{S1} \rightarrow U_{C1} \rightarrow U_{S2} \rightarrow \cdots \rightarrow U_{C3} \rightarrow U_{S4} \rightarrow U_{C4}.$$

3.2 Training algorithm

We use a fairly standard framework for our evolutionary algorithm, as shown in figure 2. The encoding strategy and genetic operators allow the optimisation of real-valued parameters and will be described in the following subsections.

3.2.1 Representation

As each parameter or weight is a real number, we use the floating-point representation, i.e., each individual is a real vector and every component of these vectors is a real number which represents a weight or a parameter.

```
PROCEDURE Evolutionary Algorithm
begin
   t := 0;
   initialise population P(t);
   evaluate P(t);
   while ( not (termination-criterion) ) do
   begin
      t := t + 1;
      select P(t) from P(t − 1);
      recombine P(t);
      evaluate P(t);
   end
end.
```

Figure 2: The basic framework of our evolutionary algorithm.

Therefore, the vector has m components if there are m weights and parameters to be optimised in a training stage.

3.2.2 Initialisation

We preset the acceptable range for each parameter according their known characteristics. After these intervals have been decided, the initial population is generated with a uniform random distribution of parameters in their appropriate ranges.

3.2.3 Evaluation

The objective function we attempt to minimize by evolution is simply the squared sum of the difference between the actual outputs of the detectors in a layer, and their target (desired) outputs, accumulated over all training patterns.

If ω is an individual in the population, and $p_1, p_2, \cdots, p_n$ are the training patterns and $t_1, t_2, \cdots, t_n$ are the corresponding target output arrays (where a 1 indicates an active detector), then the objective function is defined by

$$f(\omega) = \sum_{i=1}^{n} \sum_{j=1}^{k} [o_i(j) - t_i(j)]^2,$$

where k is the number of planes in the S-layer and $o_1, o_2, \cdots, o_n$ are the corresponding actual output arrays.

3.2.4 Selection

We use linear ranking selection to reduce the dominating effect of super individuals (high fitness outliers) and avoid early convergence to local minima. Individuals in the population are selected with a probability proportional to their ranked fitness. In addition, the best individual in the population is always passed on to the new generation (the so called elitist strategy) so that for any given simulation run, the best solution arrived at is always known.

3.2.5 Recombination

Crossover and reproduction are performed with probabilities p_c and p_r respectively, with $p_c + p_r = 1$. The offspring are inserted into the new population and this procedure is repeated until the new population is full.

If reproduction is selected, the selected individual is copied directly into the new population. For crossover, two individuals from the population are chosen as parents. Keeping in mind that our genetic encoding is based on real numbers, we use the global arithmetical crossover of [Pan et al., 1998] to produce two offspring and insert them to the new population after mutation. Let $\omega_1 = (\omega_1^{(1)}, \omega_2^{(1)}, \cdots, \omega_p^{(1)})$, $\omega_2 = (\omega_1^{(2)}, \omega_2^{(2)}, \cdots, \omega_p^{(2)})$ be the two parents, and let α be a uniformly generated random number in $(0, 1)$, then the two offspring produced by the arithmetical crossover are defined by

$$\omega_1^s = (\alpha\omega_1^{(1)} + (1-\alpha)\omega_1^{(2)}, \cdots, \alpha\omega_p^{(1)} + (1-\alpha)\omega_p^{(2)}),$$
$$\omega_2^s = (\alpha\omega_1^{(2)} + (1-\alpha)\omega_1^{(1)}, \cdots, \alpha\omega_p^{(2)} + (1-\alpha)\omega_p^{(1)}).$$

Let $\omega = (\omega_1, \omega_2, \cdots, \omega_p)$ be a vector (genome) produced by crossover, then every component ω_i is selected with probability p_m for mutation. Suppose ω_k is selected, then the offspring is a vector $\omega' = (\omega_1, \cdots, \omega_k', \cdots, \omega_p)$ and ω_k' is $\omega_k + 0.1 \cdot \delta \cdot (max_k - min_k)$ or $\omega_k - 0.1 \cdot \delta \cdot (max_k - min_k)$ with equal probability. Here $\delta = \sum_{j=0}^{15} \alpha_j 2^{-j}, \alpha_j \in \{0, 1\}$ and α_j takes 1 with probability $1/16$, (min_k, max_k) is the initial interval of the component ω_k. This mutation operator is similar in spirit to the flip mutation for binary strings. It can generate any point in the hypercube with center ω defined by $\omega_i \pm 0.1 \cdot (max_k - min_k)$. But it tests more frequently those points which are closer to ω, so it favours local search. Furthermore, it is independent of location in phenotype space [Pan and Kang, 1997]. This is frequently helpful in escaping local minima.

To improve the search performance, we substitute the above fixed mutation step size of 0.1 with an adaptive parameter. If we have a successful search, the parameter is multiplied up by a factor (say 1.1) and otherwise it is multiplied down (say by 0.9).

The termination criterion is defined as: stop if the error function $f_l(\omega)$ for stage l is smaller than a given error ϵ_l, or the generation number is greater than the maximum number of generations G.

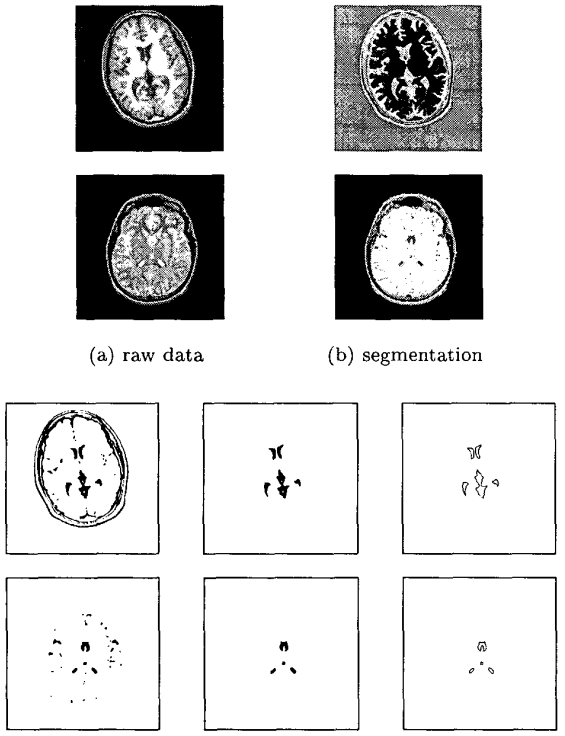
4 Experiments

In this section, we demonstrate our staged evolutionary approach by training the Neocognitron to identify com-

plex shapes in some real life images.

4.1 Description of the training and testing data set

Our experimental training and testing data are derived from magnetic resonance images of the human brain. Raw transversal image slices were acquired from 25 subjects at a resolution of 256 × 256 pixels and 256 gray levels. We trained the Neocognitron to correctly classify the outline shapes of the ventricles in these images. Image intensity invariant contour representations of this organ were obtained by pre-processing the raw data as shown in figure 3. Our method utilises an unsupervised image segmentation technique combined with rule based a priori knowledge to extract contour representations of the ventricles [Sabisch, 1998].

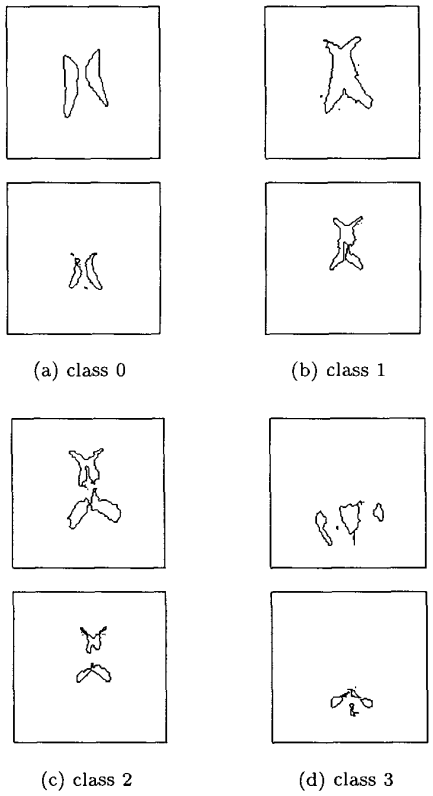


(a) raw data (b) segmentation

(c) cluster selection (d) ventricle isolation (e) contour extraction

Figure 3: Example image segmentation and contour extraction of the ventricle structure in the human brain. Two examples of the raw magnetic resonance images are given in (a). Segmented images are shown in (b). (c) A rule based system selects the intensity cluster containing the ventricle and (d) removes spurious structures. (e) The contour of the ventricle is finally obtained by boundary tracing.

160 contour images from 12 subjects were selected for training purposes, and 120 contour images from 13 subjects were retained for testing. Both data sets were manually assigned into 4 distinct classes corresponding to different volume segments of the 3D image. Thus, the training set consisted of 40 patterns per class and the testing set contained 30 patterns per class. Two example example patterns for each class are shown shown in figure 4. Because of the high computational load of the evolutionary algorithm, the contour images were reduced to a resolution of 32 × 32 pixels prior to training.



(a) class 0 (b) class 1

(c) class 2 (d) class 3

Figure 4: Example patterns for the 4 ventricle categories.

4.2 Configuration and training parameters

We utilised a Neocognitron with 4 pairs of processing layers to recognise our 32 × 32 contour patterns. The evolutionary algorithm optimised the selectivity r (see (1)), the receptive field weighting parameter of inhibitory c-cells γ (see (6)), the maximum strength of the receptive field weighting of s-cells $\bar{\delta}$ and its roll-off δ (equation (7)). Further, the EA was utilised to optimise the weights of excitory weights a and the inhibitory weight b of an S-cell (see (1)).

The topology determining parameters were kept con-

stant during the evolutionary process, i.e. for each layer of the network the number of planes (features), size of a plane, the receptive field size and the receptive field overlap are manually specified. Table 1 summarises the network configuration parameters. A cross section of the connection patterns of our implementation is shown in figure 5.

Layer	receptive field size	receptive filed overlap	no of planes	plane size
U_{S1}	5	4	10	31
U_{C1}	5	3	10	15
U_{S2}	3	2	9	15
U_{C2}	3	1	9	9
U_{S3}	3	2	4	7
U_{C3}	3	2	4	7
U_{S4}	3	2	4	5
U_{C4}	5	4	4	1

Table 1: The network configuration parameters exploited during the evolution of the training parameters of the network.

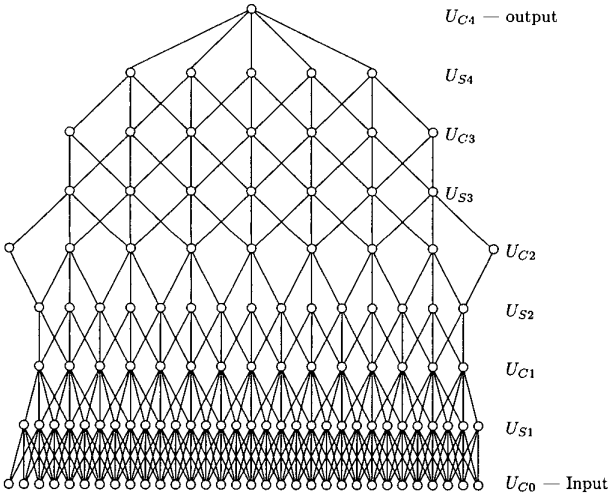


Figure 5: The connectivity of a Neocognitron.

The evolutionary algorithm is controlled by several parameters which are summarised in table 2.

4.3 Experimental Results

For the classification task described above, it was reported in [Sabisch, 1998] that the Neocognitron could not achieve a reasonable recognition rate of the training set using Fukushima's competitive learning algorithm. However, our staged evolutionary algorithm achieved a mean recognition rate of 70.8% with a standard deviation of only 0.059 over 10 random runs, as shown in table 3. Figure 6 shows example training patterns we

parameter	symbol	value
population size	N	100
crossover prob	p_c	0.9
reproduction prob	p_r	0.1
mutation prob	p_m	0.01
max generations	G	10^4
initial weight limits	$w_{min} \cdots w_{max}$	$0 \cdots 1$

Table 2: Control parameters exploited in our evolutionary algorithm.

used to train the first two stages. These patterns were extracted from the input images manually. For the third stage, the training patterns consisted of the input images cropped by 3 pixels on 2 sides. Figure 7 and 8 shows the training performance and the evolution of recognition rate of the best individual in the population for a run, respectively.

Maximum	Average	Minimum	Standard deviation
80.7	70.8	62.7	0.059

Table 3: The recognition rate of a Neocognitron.

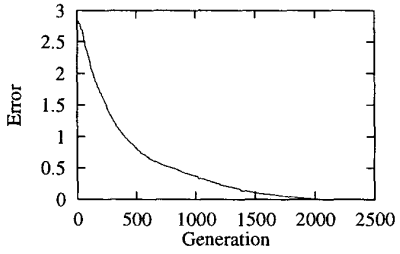
(a) 10 training features with resolution 5×5 for the first stage.

(b) 9 training features with resolution 13×13 for the second stage.

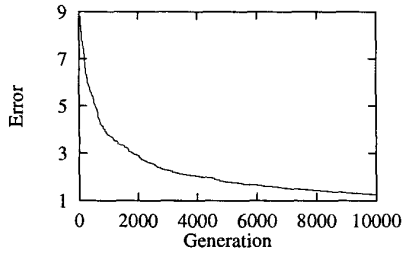
Figure 6: Training features for the first two stages.

It has been claimed that a high learning rate ($\approx 10^4$) should be used to train a Neocognitron by competitive learning [Lovell, 1994]. This will lead to large weights in the Neocognitron. However, large weights will cause excessive variance of the output and hurt the generalisation ability of neural networks. In our approach, the initial weight domain of Neocognitron is $[0, 1]$ and the weights finally fall into a small interval such as $[-5, 5]$ in our experiments.

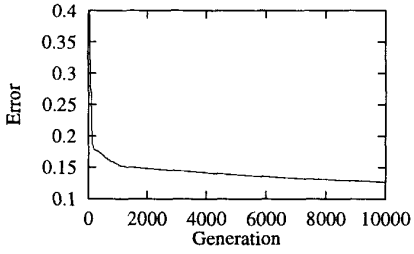
Other architectures such as the well-known Multi-Layer Perceptron (MLP) have also been applied to the same test problem [Sabisch, 1998]. For the image size used during this set of experiments the Neocognitron architecture did not outperform an equivalent Multi-Layer Perceptron. However, for large image sizes the performance of the MLP degraded severely, whereas the hi-



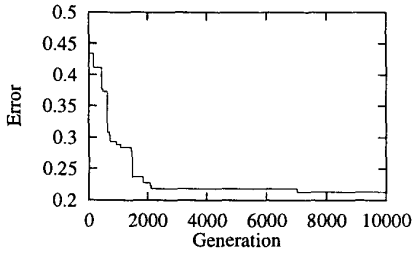
(a) first stage



(b) second stage



(c) third stage



(d) final stage

Figure 7: Performance of training the four stages by EA.

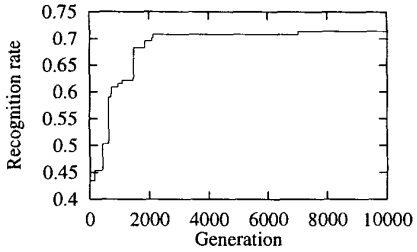


Figure 8: The evolution of the recognition rate.

Stage l		1	2	3	4
U_{Sl}	r_l	0.42	0.20	1.35	0.20
	γ_l	0.77	0.95	0.80	0.84
U_{Cl}	δ_l	0.34	0.30	0.25	0.44
	$\bar{\delta}_l$	0.39	0.68	0.39	1.53

Table 4: The best set of evolved training parameters of the Neocognitron which resulted in a recognition rate of 80.7%.

erarchical arrangement offered improved scalability and better tolerance towards pattern distortion, translation and scaling. This feature is a direct consequence of the weight sharing causing a natural regularisation in the network. Our configuration of the Neocognitron requires 1571 weights, whereas a fully connected MLP with only one hidden layer of 10 hidden nodes has 10280 connections, approximately an order of magnitude more. The number of weights in the Neocognitron does not grow with the image size unless additional layers are required. Even in that case the increase with image size is only approximately linear in the Neocognitron, while a quadratic increase is observed in the MLP network. For large image sizes, the difference could be several orders of magnitude.

Results presented in the literature as well as our experiments show that skillful feature selection improves the recognition rate [Fukushima and Wake, 1991]. However, manual feature selection can not be applied to large and complex pattern sets efficiently. Unfortunately, Fukushima & Miyake's self-organised version of the Neocognitron [Fukushima and Miyake, 1982] typically has even poorer generalisation performance. Elsewhere [Sabisch et al., 1998], we have reported the design of a Neocognitron-like hierarchical neural network which uses Self-Organising Maps to extract the features automatically. Description of our modifications to the Neocognitron falls beyond the scope of this paper, but the network has been demonstrated to exceed the performance of fully-connected MLPs, Radial Basis Function networks, and Higher Order neural networks by at least 10%.

5 Conclusions

The Neocognitron, a recognition system mimicking the mammalian visual architecture, has been widely acclaimed for its shift and deformation tolerance. But it is a complex neural network with numerous parameters and weights which need to be optimised for each application. The Neocognitron used for the experimental work in this paper has 1571 weights and parameters. It is not easy to optimise these parameters and weights by gradient decent algorithms.

We presented a staged training approach using evolutionary algorithms that successfully meets this challenge when applied to real world problems.

The Neocognitron was first proposed before perceptrons were popular. The mechanism of its S-cells is similar to a perceptron's. But the computational cost of a perceptron is much lower than that of an S-cell. For this reason, a number of researchers [Sung and Wilson, 1990, Lovell, 1994, Sabisch et al., 1997] have proposed similar hierarchical neural network architectures, which use multiple perceptrons to replace the S-planes of the Neocognitron. However, because of the difficulty of training Neocognitrons, it has not been possible to compare the Neocognitron with such 'competitors'. With the success of our staged evolutionary training methodology for the Neocognitron, such comparisons are now possible.

Bibliography

[Abutaleb et al., 1989] Abutaleb, A., Delalic, Z. J., Ech, R., and Siegel, J. A. (1989). Automated analysis for scintigraphic evaluation of gastric emptying using invariant moments. *IEEE Transactions on Medical Imaging*, 8(4):364–369.

[Chin and Dyer, 1986] Chin, R. T. and Dyer, C. R. (1986). Model-based recognition in robot vision. *Computing Surveys*, 18:68–108.

[Fukushima, 1988] Fukushima, K. (1988). Neocognitron: a hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1:119–30.

[Fukushima and Miyake, 1982] Fukushima, K. and Miyake, S. (1982). Neocognitron: a new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*, 15(6):455–469.

[Fukushima and Wake, 1991] Fukushima, K. and Wake, N. (1991). Handwritten alphanumeric character recognition by the neocognitron. *IEEE Transactions on Neural Networks*, 2(3):355–65.

[Hinton, 1987] Hinton, G. E. (1987). Learning translation invariant recognition in a massively parallel network. In *Proceedings of Parallel Architectures and Languages Europe (PARLE)*, pages 1–13.

[LeCun and Farber, 1989] LeCun, Y. and Farber, R. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

[Lovell, 1994] Lovell, D. R. (1994). *The Neocognitron as a system for handwritten character recognition: limitations and improvements.* PhD dissertation, University of Queensland.

[Lovell et al., 1997] Lovell, D. R., Downs, T., and Tsoi, A. C. (1997). An evaluation of the neocognitron. *IEEE Transactions on Neural Networks*, 8(5):1090–105.

[Pan and Kang, 1997] Pan, Z. and Kang, L. (1997). An adaptive evolutionary algorithms for numerical optimization. In Yao, X., Kim, J. H., and Furuhashi, T., editors, *Simulated Evolution and Learning*, number 1285 in Lecture Notes in Computer Science, pages 27–34, Heidelberg. Springer-Verlag.

[Pan et al., 1998] Pan, Z., Kang, L., and Chen, Y. (1998). *Evolutionary Computation.* Tsinghua University Press, Beijing (In Chinese).

[Sabisch, 1998] Sabisch, T. (1998). *Towards automatic registration of magnetic resonance images of the brain using neural networks.* PhD dissertation, University of Hertfordshire, UK.

[Sabisch et al., 1997] Sabisch, T., Ferguson, A., and Bolouri, H. (1997). Rotation, translation and scaling tolerant recognition of complex shapes using a hierarchical self-organising neural network. In *Proceedings of international conference on neural information processing (ICONIP'97)*, volume 2, pages 1174–1178, Dunedin, New Zealand. Springer.

[Sabisch et al., 1998] Sabisch, T., Ferguson, A., and Bolouri, H. (1998). Identification of complex shapes using a self-organising neural system. *IEEE Transactions on Neural Networks.* submitted Oct. 1998, (TNN-A245).

[Shridhar and Badreldin, 1984] Shridhar, M. and Badreldin, A. (1984). High accuracy character recognition algorithm using fourier and topological descriptors. *Pattern Recognition*, 17(5):515–524.

[Sung and Wilson, 1990] Sung, C. and Wilson, D. (1990). Percognitron: neocognitron coupled with perceptron. In *International Joint Conference on Artifical Neural Networks*, pages 753–758, Ann Arbor, USA. IEEE.

[Teo et al., 1997] Teo, M. Y., Khoo, L. P., and Sim, S. K. (1997). Application of genetic algorithms to optimise neoconitron network parameters. *Neural Network World*, 7(3):293–304.