Pergamon

Neural Networks, Vol. 7, Nos. 6/7, pp. 985–1004, 1994

0893-6080/94 $6.00 + .00

0893-6080(94)E0031-F

1994 SPECIAL ISSUE

Design and Evolution of Modular Neural Network Architectures

BART L. M. HAPPEL AND JACOB M. J. MURRE

Leiden University, Unit of Experimental and Theoretical Psychology

(*Received* 28 *September* 1993; *revised and accepted* 9 *February* 1994)

Abstract—*To investigate the relations between structure and function in both artificial and natural neural networks, we present a series of simulations and analyses with modular neural networks. We suggest a number of design principles in the form of explicit ways in which neural modules can cooperate in recognition tasks. These results may supplement recent accounts of the relation between structure and function in the brain. The networks used consist of several modules, standard subnetworks that serve as higher order units with a distinct structure and function. The simulations rely on a particular network module called the categorizing and learning module. This module, developed mainly for unsupervised categorization and learning, is able to adjust its local learning dynamics. The way in which modules are interconnected is an important determinant of the learning and categorization behaviour of the network as a whole. Based on arguments derived from neuroscience, psychology, computational learning theory, and hardware implementation, a framework for the design of such modular networks is presented. A number of small-scale simulation studies shows how intermodule connectivity patterns implement "neural assemblies" that induce a particular category structure in the network. Learning and categorization improves because the induced categories are more compatible with the structure of the task domain. In addition to structural compatibility, two other principles of design are proposed that underlie information processing in interactive activation networks: replication and recurrence.*

Because a general theory for relating network architectures to specific neural functions does not exist, we extend the biological metaphor of neural networks, by applying genetic algorithms (a biocomputing method for search and optimization based on natural selection and evolution) to search for optimal modular network architectures for learning a visual categorization task. The best performing network architectures seemed to have reproduced some of the overall characteristics of the natural visual system, such as the organization of coarse and fine processing of stimuli in separate pathways. A potentially important result is that a genetically defined initial architecture cannot only enhance learning and recognition performance, but it can also induce a system to better generalize its learned behaviour to instances never encountered before. This may explain why for many vital learning tasks in organisms only a minimal exposure to relevant stimuli is necessary.

Keywords—Neural network architectures, Structural compatibility, Replication, Recurrence.

1. INTRODUCTION

The human brain is the most elaborate information processing system known. In current thinking, it derives most of its processing power from the huge numbers of neurons and connections. Our brain contains about 10^{11} neurons, each of which is connected to an average of 10^4 other neurons. This amounts to a total of 10^{15} connections. If these billions of connections were fully random, it can be shown that the brain would be many times larger than it actually is (Murre, 1993; Nelson & Bower, 1990). It is, indeed, only because of its highly organized architecture that the brain manages to execute a myriad of functions and yet maintains a compact size. Implementation is, however, only one constraint. In this paper we will argue that the structure of the brain has evolved to capture as many regularities of the world around us as possible. We will show how neural structures not only allow for rapid and efficient learning, but also enable a system to better generalize its learned behaviour to new instances. These neural architectures, found in nearly all organisms, reflect in a profound way the structure of patterns and tasks essential for survival.

A principal form of global organization of the brain is its layered structure. Subsequent layers of neurons,

Acknowledgement: This work is sponsored in part by the Dutch Foundation for Neural Networks and by the Medical Research Council.

Requests for reprints should be sent to Bart L. M. Happel, Leiden University, Unit of Experimental and Theoretical Psychology, P.O. Box 9555, 2300 RB Leiden, The Netherlands. E-mail: Happel@rulfsw.leidenuniv.nl. The second author is presently at the MRC Applied Psychology Unit, 15 Chaucer Road, Cambridge CB2 2EF, United Kingdom. E-mail: Jaap.Murre@mrc-apu.cam.ac.uk

arranged in a hierarchical fashion, form increasingly complex representations. Such multistage information processing can, for example, be identified in the primary visual system (Hubel & Wiesel, 1959, 1962, 1965). Apart from a layered, hierarchical structure in the primate brain, we find multiple parallel processing streams or pathways that constitute another major neural construction principle (e.g., Livingstone & Hubel, 1988; Kosslyn, Flynn, Amsterdam, & Wang, 1990; Van Essen, Anderson, & Felleman, 1992; Zeki & Shipp, 1988). Structuring of the brain into distinct streams allows for the independent processing of different information types and modalities. A good example can also be found in the visual system, where different aspects of visual stimuli such as form, color, motion, and place are processed in parallel by anatomically separate neural systems, organized in the magnocellular and parvocellular pathway (e.g., Livingstone & Hubel, 1988; Hilz & Rentschler, 1989). Mediated by convergent structures (Zeki & Shipp, 1988), the separately processed visual information is integrated at higher levels to produce a unitary percept (e.g., Poggio, Gamble, & Little, 1988; DeYoe & Van Essen, 1988).

In addition to a global organization of the brain into layers and streams, we also find a highly regular structure at the more microscopic level. Modules containing little more than a hundred cells, also known as minicolumns (Mountcastle, 1975), have been proposed as the basic functional modular unit of the cerebral cortex (Mountcastle, 1975; Szentágothai, 1977; Eccles, 1981). Although little is known about the exact working of the cortex, Creutzfeldt (1977) suggests that it functions as a cooperative network in which all areas are subject to the same general structural principles. The specific function of any cortical area is largely defined by the origin of its afferents and the destination of its efferent connections. In the visual system, for example, it has been worked out in some detail how the interaction between these modules may result in the emergence of the clever mechanisms of vision (e.g., Bülthoff & Mallot, 1987; Cavanagh, 1987; Terzopoulos, 1986).

We propose to capture these global and local structural regularities by modelling the brain in a framework of modular neural networks (Murre, 1992a; Murre, Phaf, & Wolters, 1989, 1992). Rather than starting from a fully connected network and then letting the desired functions emerge with prolonged learning, we advocate an approach that proceeds from a structured, modular network architecture. Such networks possess an initial architecture that consists of many modules. Whereas within-module connections may be dense, the modules themselves are only sparsely interconnected. These intermodular connections define the initial network architecture. By choosing the right modular structure, networks with a wide range of desirable characteristics can be developed, as will be outlined below.

We will first argue that a basic neural network module can be derived from constraints provided by neurophysiology, experimental psychology, and from characteristics imposed by implementation and information processing. Then, we will present a case study in which we explain the behaviour of some small modular networks. Finally, we will proceed to show how a method gleaned from natural evolution, called genetic algorithms, may be applied to the design of architectures that possess an increased ability to learn and to generalize the learned behaviour to problem instances not encountered previously.

2. NEURAL ARCHITECTURE AND FUNCTION

2.1. Modularity

As was briefly sketched above, several lines of evidence indicate that a modular organization of the brain exists at different anatomical scales. This modular organization is paralleled at the functional level of information processing, as has become evident from a wide range of psychological studies. The anatomical division of the brain into the major hemispheres, for example, is paralleled at the functional level by hemispheric specialization. Whole groups of mental functions are allocated to different halves of the brain. Split brain patients in which the connection between the two hemispheres is completely severed can live an almost normal life, which shows that the hemispheres indeed function to a large extent independently. It has been known for nearly a century that within each hemisphere individual functions are organized into anatomically separate regions (Brodmann, 1909). Analysis of behavioural functions indicate that even the most complex functions of the brain can be localized at least in a broad manner (e.g., Kandel & Schwartz, 1985; Posner, Petersen, Fox, & Raichle, 1988). Studies of localized brain damage, for example, reveal that isolated, mental abilities can be lost as a result of local lesions, leaving other abilities unimpaired (e.g., Damasio, Damasio, & Van Hoesen, 1982; Warrington, 1982). Also, different types of aphasia and related language disorders indicate that different functions are separately localized in the brain.

A functional advantage of the anatomical separation of different functions might be the minimization of mutual interference between simultaneous processing and execution of different complex tasks. Most of us, for example, have no problems in driving a car while listening to the radio, at the same time and without much interference. Studies with multiple-task execution indicate that some tasks can easily be performed in parallel, while the simultaneous execution of similar tasks (e.g., presentation of two auditory or two visual messages) causes much more interference. The differ-

ence in performance found in these tasks can be explained by assuming a modular organization of the brain (e.g., Allport, 1980). Some tasks are processed in distinct streams of modules and do not interfere with each other. Other tasks require simultaneous access to a single module and are, therefore, much more prone to mutual interference.

Individual functions can be further subdivided into functionally different subprocess or subtasks. Often these more specialized functions can be localized into anatomically separate regions. For example, it has been demonstrated that the human information processing system uses a number of subsequent levels of encoding in performing lexical tasks (Posner, 1986; Marshall & Newcombe, 1973). Positron emission tomography (PET-scans) studies show that these functionally distinguished subprocess are separately localized in the brain (Posner et al., 1988).

An important argument that can be derived from these studies is that the nature of information processing in the brain is modular. Individual functions are broken up into subprocesses that can be executed in separate modules without mutual interference. We could speculate that the subdivision of modules into smaller modules and of functions into subfunctions, might continue until extreme depth. Some of these "microfunctions" may be directly localized in specific parts of the cortex. As mentioned above, modules containing only a few hundred cells, also known as minicolumns (Mountcastle, 1975), have been proposed as the basic functional and anatomical modular units of the cerebral cortex (Mountcastle, 1975; Szentágothai, 1977; Eccles, 1981). These modules are thought to cooperate in the execution of cortical functions (Creutzfeldt, 1977). As the classic experiments by Hubel and Wiesel (1959, 1962, 1965) illustrate, even single neurons in such modules can be found to underlie certain microfunctions, such as "detecting a short line moving at a certain speed in a certain direction at a certain position on the retina."

The modular organization of the brain might form the necessary neural substrate for decomposing tasks and the independent processing of subtasks, as well as allow for specific interactions between these subprocesses. The architecture of the brain is the result of a long evolutionary process during which a large set of specialized subsystems emerged interactively carrying out the tasks necessary for survival and reproduction (e.g., Gazzaniga, 1989). Learning in biological neural systems, in our view, serves as a mechanism for fine tuning these broadly laid out neural circuits. Or to put it differently, learning is directed by the initial architecture of the brain. Although it is improbable that the genes code all structural information about the brain, they may be the ultimate determinant of what mental functions can and cannot be learned (e.g., Changeux & Danchin, 1976).

2.2. Structure in Connectionist Models

One of the concerns of this study is the design of artificial learning neural network architectures. The natural neural system provides some valuable leads for the design of networks that have superior performance (e.g., see Shepherd, 1990, and Shepherd & Koch, 1990, for an overview of the computational capacities of synaptic circuits). We see a parallel in the design of neural network architectures to solve practical tasks and the design of the brain. It will be argued that imposing initial structural constraints, similar to those of the brain, on artificial neural network architectures provides a powerful means to overcome important limitations of connectionist approaches to solving complex real-world problems. The strengths and weaknesses of different architectures as encountered in computational simulations may, in turn, shed light on the processing principles underlying the architecture of the brain.

Many of the currently popular connectionist models rely on little initial structure. Some networks assume total interconnectivity between all nodes (e.g., Hopfield, 1982). Others assume a hierarchical, multilayered structure (e.g., Rumelhart, Hinton, & Williams, 1986), in which each node in a layer is connected to all nodes in neighbouring layers. The advantage of such fully connected systems is that they are extremely plastic. Given enough resources (i.e., hidden nodes), any input–output mapping can be encoded in the weights (Funahashi, 1989; Hornik, Stinchcombe, & White, 1989, 1990). Whereas for many situations this is a desirable characteristic, it appears that *learning* a large-scale task "from scratch" in such networks may take a very long time. The number of iterations necessary for a network to reach convergence increases with the size of the network. But, more importantly, hardware implementation of such large systems becomes problematic very soon. Implementing, for example, a Hopfield network with 10,000 nodes means that at each iteration 10^8 connections must be processed. Due to the inordinately high communication overhead, even implementation in parallel computers may not be very beneficial (Murre 1992a; also see Murre et al., 1992).

A suitable theoretical perspective that sheds some light on the relation between architecture and function of neural networks is provided by Solla (1989). She describes how a learning neural network model defines a probability distribution over the space of its possible input–output mappings. The shape of this distribution is an intrinsic property of the architecture and the dynamics of the network under consideration. The entropy of the distribution is defined as a measure of the diversity of the input–output mappings realizable by the network. Learning from examples can be viewed as a method to reduce the intrinsic entropy by excluding network configurations (i.e., certain combinations of weight values on the internal connections) that realize

mappings incompatible with the training set. Effective rule extraction from examples must be directed at finding mappings compatible with the *entire* task domain, rather than just with the training examples encountered. Such mappings are said to generalize well (from training set to task domain). In general, effective rule extraction is likely to occur, if the summed intrinsic probability of all network configurations that realize the desired mapping is high. If the architecture prohibits the formation of undesired mappings, learning is greatly facilitated and the network will generalize well.

It can be argued that fully connected neural networks only represent a very limited subset of possible network architectures. Problems with such networks result from specific intrinsic properties, leading to unfavorable characteristics of the probability distribution of network configurations. Because this probability distribution, and thus the functional capabilities of the network, is determined both by the network dynamics and the architecture, it can be influenced in two ways: by changing the dynamics or by changing the network architecture. Although recently several studies have appeared that focus on other methods like "growing neural networks" (e.g., Frean, 1990; Fritzke, 1991), emphasis in connectionist modelling has long been on the first approach in the form of the development of more efficient learning algorithms. Such attempts, however, have necessarily been limited to the adjustment of a small number of relatively fixed formalisms like delta-rule learning and Hebbian learning in fully connected architectures. A potentially much more versatile method of changing the probability distribution of network configurations is to constrain the network topology. Theoretically, there is virtually no limit to the possible ways in which this distribution can be altered by imposing constraints on connectivity. Consequently this should provide a sufficient increase in the learning capacities of neural networks to apply them to large-scale problems. A major problem with this approach, however, is that there currently exist no general methods or guidelines providing useful architecture constraints. This may be one of the reasons why entropy reduction was largely left to a learning process in unstructured networks. The third part of this article will describe an approach that addresses this problem.

Neural network studies show that in some specific cases fairly straightforward topological constraints can be found that improve learning and generalization. Rueckl, Cave, and Kosslyn (1989) demonstrated that in learning to encode both form and place of visual stimuli, a standard three layer back propagation network learns faster and makes less mistakes when the hidden layer is split in such a way that separate network resources (modules) are dedicated to the processing of "what" and "where." In a study of Solla (1989), it is shown that in solving the "contiguity problem," small receptive fields form "edge-detectors" that greatly facilitate learning this specific task.

Some studies show that constraints on the architecture need not always be specific. For example, random pruning of connections before any learning has taken place may improve network performance (Barna & Kaski, 1990). Several researchers have commented on the advantages of limiting architectures. LeCun et al. (1990), for example, state that "good generalization can only be obtained by designing a network architecture that contains a certain amount of a priori knowledge about the problem. The basic design principle is to minimize the number of free parameters that must be determined by the learning algorithm, without overly reducing the computational power of the network. This principle increases the probability of correct generalization because it results in a specialized network architecture that has a reduced entropy."

In addition to network entropy several other measures have been developed to capture the complexity (or simplicity) of a network architecture and to predict its generalization behaviour (e.g., Baum & Haussler, 1989; Blumer, Ehrenfeucht, Haussler, & Warmuth, 1989; Haussler, 1990; Wolpert, 1990; also see Vailliant, 1984). These can be called measures of simplicity. They characterize a network's expressiveness or its number of free parameters. All of the measures developed are related to the principle of parsimony generally applied to scientific theories and commonly known as Occam's Razor (Wolpert, 1990). The major conclusion of the above research is that simple networks generalize better. As we will show below there exist other ways of improving generalization. Later in this article we will present a case where apparently more complex architectures, in fact, learn and generalize better than simple ones.

Imposing constraints on neural network architectures can, thus, improve their performance. The particular problem that will be addressed in this study is how to find suitable structural constraints for neural networks dedicated to specific tasks. Modular constraints on connectivity, as found in the brain, may provide useful guidelines for the effective design of artificial neural network architectures. Modularity will, therefore, be a main theme throughout this study. The experiments presented below are all based on a specific neural network model, called the categorizing and learning module (CALM) (Murre, 1992a; Murre et al., 1989, 1992). The methods and principles of modular network design set forth in this study, however, are of a more general nature and can be applied to other learning neural network formalisms. After having introduced CALM in the next section and after having explained the behaviour of some multimodule CALM networks, we will describe in the final section of this article how a search method based on an evolution

metaphor can be applied as an automatic optimization procedure for modular neural network architectures. Some of the solutions found by this artificial evolution method mimic the structures found in the brain in surprising ways.

3. CATEGORIZATION AS A COMPUTATIONAL PRIMITIVE

A connectionist model consists of many interconnected, autonomous basic elements. Each of these elements itself only has a simple function. Complex functions are obtained as emergent behaviour of the dynamic interactions of the simple elements. In the biological system, even elements as low as the synapse exhibit a sufficiently complex computational behaviour to take these, rather than the neuron, as basic computational units (Shepherd, 1990). But because we are more interested here in investigating the broad computational capacities of neural networks we shall take the opposite direction. When instead of a single artificial neuron (node), we take an entire module (consisting of many nodes) as a computational unit, we may expect more elaborate behaviour to emerge from aggregations of interacting modules. A module, in this sense, can be said to form a higher order functional element.

The autonomy of functional elements in dynamic systems like neural networks, seems to be an important prerequisite for the emergence of complex behaviour. In connectionist systems, autonomous functioning can be translated into functioning without the aid of external supervision. Neural elements that can function without supervision must be able to autonomously discriminate between, and respond differently to, inputs that are dissimilar according to a specified criterion or evaluation function. A single artificial neuron typically evaluates its input as a sigmoid function of the summed incoming activation. Higher order functional elements, such as modules, allow for more elaborate, autonomous input evaluations, resulting in more elaborate behaviour emerging from their collective interactions.

The simulations reported below take CALM as a basic computational unit. A detailed description can be found in Murre (1992a), Murre et al. (1989, 1992), and Phaf (1992). CALM is a modular, network algorithm that has been especially developed as a building block for modular interactive neural networks. The most important feature of a CALM is its ability to autonomously categorize input activation patterns into discrete categories. The use of CALMs introduces categorization as a computational primitive in neural networks for modelling complex behaviour. The concept of pattern recognition and categorization as a fundamental process in cognition has a long history in psychology (for a review, see Wickelgren, 1981) and can be seen to supplement the strictly associative accounts that have an even longer history that may be traced back to antiquity (Anderson & Bower, 1974). Incorporation of such psychological concepts into neural networks may help to mitigate practical problems with current associative models. In the next subsection, the way categorization is operationalized in the CALM model will be described briefly. The behaviour of a single module will be illustrated by introducing a small categorization task. The same task will then be used in the following subsection to illustrate some processing principles resulting from the interaction of modules in multimodule networks.

3.1. Categorization in Single-Module CALM Networks

A number of architecture constraints in CALM rely on the general architecture of the neocortical minicolumn (Szentágothai, 1975). These neuroanatomical modules consist of a variety of cell types. A major division can be made into *excitatory* pyramidal cells, which form long-ranging connections to other cortical regions, and various types of *inhibitory* interneurons (e.g., basket cells), which make short-range within-module connections. The structural principle of *intramodular inhibition* implies that the main process within a module will be a form of competition. Competition is a powerful mechanism that has been used to simulate a large number of psychological and neurophysiological phenomena (e.g., Bridgeman, 1971; Cornsweet, 1970; Fukushima, 1975, 1988; Grossberg, 1976a, 1982, 1987; McClelland & Rumelhart, 1981; Phaf, Van der Heijden, & Hudson, 1990; Rumelhart & Zipser, 1985; Von der Malsburg, 1973; Walley & Weiden, 1973). It enables a system to autonomously categorize patterns (Grossberg, 1976a) and can, therefore, be used to implement unsupervised competitive learning.

Unsupervised autonomous information processing, in particular categorization and learning, are the principal functions of the CALM model. A schematic overview of the connection structure underlying these functions is shown in Figure 1. A CALM consists of a number of *representation nodes* (R-nodes) that are fully connected with distant input nodes through modifiable connections. The R-nodes are the only nodes that form connections with nodes outside the module (either sending or receiving). Extramodular connections to R-nodes are always modifiable. All other connections, inside the CALM, are nonmodifiable (i.e., the architecture of the module itself remains fixed). The input to a CALM can either originate from another CALM or from an activation pattern provided through a specialized input device that will be called an *input module.* The number of R-nodes in a CALM will be referred to as its *size.* Categorization in a CALM is operationalized as the association of an input pattern with a

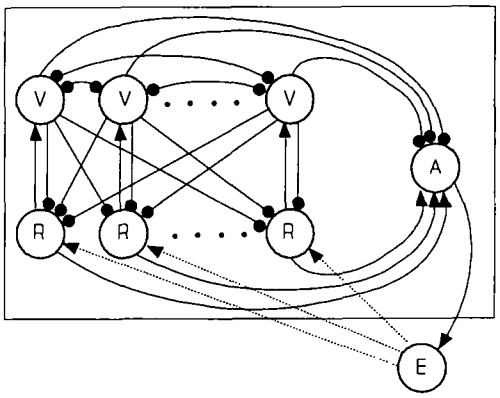

FIGURE 1. The internal wiring scheme of a CALM module. The learning weights are not shown. V, R, A, and E denote the four node categories: veto-, representation-, arousal-, and external-nodes.

unique R-node that is then said to represent the pattern. CALM uses continuous activation values between 0.0 and 1.0 that are iteratively updated according to a sigmoid activation function.

Categorization proceeds through the resolution of an interactive, winner-take-all competition between all R-nodes activated by the input. Initially, all learning weights are set to equal values (i.e., they are typically not initialized by setting them to random values). Consequently, at the very first presentation of a pattern to a CALM, all R-nodes are equally activated. This results in a state of maximal competition. The resolution of competition (the *convergence* process) is mediated by specialized inhibitory *veto nodes* (V-nodes) and a *state-dependent* noise mechanism. The noise mechanism distributes random activations over all R-nodes during and immediately after the competition process, helping the module to escape from competition deadlocks. It also helps to find more optimal overall categorizations (deeper attractors) in a manner akin to the Boltzmann machine (Ackley, Hinton, & Sejnowski, 1985). The amplitude of this noise is made proportional to the amount of competition, as measured through the number of coactive R-nodes by the so-called *arousal node* (A-node, see Figure 1). The A-node is connected to an *external node* (E-node), which distributes the random activations over the R-nodes.

During and following the categorization process a form of Hebbian learning takes place that preserves input–output associations by adjusting the learning weights from the input nodes to the R-nodes of a module. The properties of a newly formed category are defined by the weights of the connections from the input to the R-node representing that category. Different, subsequently presented input activation patterns will lead to different representations. The learning rule used in CALM is derived from Grossberg (1976b) with some modifications. The Grossberg rule enables a module to discriminate also between nonorthogonal patterns. The amount of learning (as is the noise level) in CALM is made proportional to the level of competition as measured by the A-node activation.

Learning in CALM presents a possible solution to the *plasticity-stability dilemma* in neural network modelling (Carpenter & Grossberg, 1987, 1988). This problem addresses the tension between the two counteracting constraints of a sufficiently plastic structure to accommodate new patterns, while simultaneously ensuring the stability of existing representations. The approach taken here, where the learning rate is made dependent on the relative novelty of the input pattern, implements an "elaboration-activation learning mechanism." Graf and Mandler (1984; Mandler, 1980) proposed a subdivision of learning processes to explain a major dissociation of memory observed in a wide variety of experimental tasks (i.e., implicit and explicit memory tasks, see Murre, 1992a, Phaf 1991, and Phaf, Postma, & Wolters, 1990 and submitted, for a more thorough exposition of the theoretical motivation of CALM). The presentation of a new pattern to a CALM module will result in much competition. The ensuing exploration and establishment of a new category will be facilitated by the distribution of high-amplitude noise and a high learning rate (elaboration type learning). The presentation of a pattern already learned will result in much less competition, a small amount of noise, and only base rate learning (activation type learning). This implementation of two distinct learning processes can also be related to Hebb's (1955) ideas

TABLE 1
(a) Eight Binary Patterns and (b) Their Squared Euclidian Distances

p_1	1	1	1	1	0	0	1	0
p_2	1	1	1	1	0	1	0	0
p_3	1	1	1	0	1	1	0	0
p_4	1	1	0	0	1	1	1	0
p_5	0	0	1	1	1	0	0	1
p_6	0	0	0	1	1	0	1	1
p_7	0	0	0	1	0	1	1	1
p_8	0	0	1	0	0	1	1	1

(a)

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
p_1	0	2	4	4	5	5	5	5
p_2		0	2	4	5	7	7	5
p_3			0	2	5	7	7	5
p_4				0	5	5	5	5
p_5					0	2	4	4
p_6						0	2	4
p_7							0	2
p_8								0

(b)

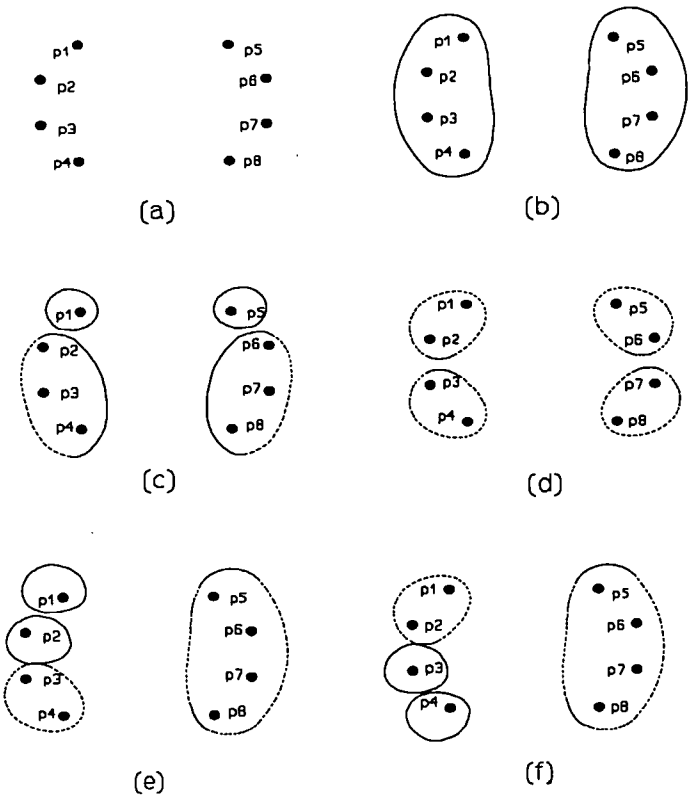

FIGURE 2. Graphical representation of (a) the stimulus space and categorizations performed by single CALM modules of sizes (b) 2 and (c–f) 4.

about the role of nonspecific arousal effects in the encoding of stimuli (hence, the name *arousal* node).

3.1.1. *A Simple Experiment in Categorization.* In general, categorization in CALM implies both discrimination of dissimilar input patterns and clustering of similar input patterns. In discrimination, the representational space is divided into nonoverlapping regions. The process of dividing the representational space, however, cannot go on indefinitely. A module tends to cluster stimuli by grouping similar inputs within the same regions.

To illustrate the autonomous categorization behaviour of a single CALM, a small categorization task was devised, involving eight nonorthogonal, binary patterns, that had to be categorized without supervision. (It should be noted that CALM is also able to handle continuous input activation patterns.) The eight patterns ($p_1, \ldots, p_8$) are shown in Table 1a. The squared euclidian or Hamming distance is shown as a measure of similarity between the patterns in Table 1b. A two-dimensional, approximate, graphical representation of the relative distances between the eight-dimensional stimuli is shown in Figure 2a. It can be seen that the patterns can be divided in two main clusters: one containing patterns p_1, p_2, p_3, and p_4 and the second containing patterns p_5, p_6, p_7, and p_8. These clusters are defined by the fact that the distance between any two patterns within the same cluster is smaller than the distance between any two patterns from different clusters. Patterns from different main clusters were presented in an alternating order, $p_1, p_5, p_2, p_6, p_3, p_7, p_4, p_8$, to a single-module architecture containing an input module connected to a single CALM (see Figure 3). One stimulus presentation lasted 100 iterative updates of the learning weights and activations. Convergence was always reached well within the duration of a stimulus presentation. In between two stimulus presentations all network activations were reset to zero. Learning weights were never reset. During the categorization process, shifts in stimulus representations may occur due to the stochastic nature of categorization in CALM,

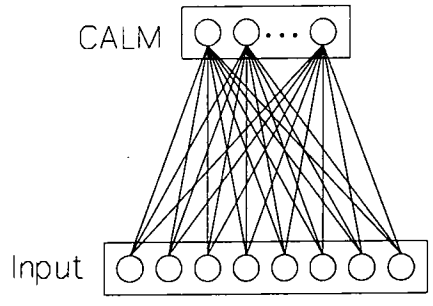

FIGURE 3. Architecture of a model with a single CALM module. The forward connections are modifiable. Only the R-nodes of the CALM module are shown.

and due to changes in initial category properties resulting from subsequent stimulus categorizations. The eight patterns were, therefore, presented a number of times (in the order listed above) until a stable categorization had been reached.

When presented to a CALM of size 2, the patterns are divided into two categories in accordance with their Hamming distances. Patterns of the same cluster are represented on the same R-node (see Figure 2b). Larger modules make finer discriminations in their input. For example, a module of size 4 will allocate the input patterns to 4 categories. After a single presentation of the eight patterns, it was typically found that p_1 and p_5 formed two separate categories, while p_2, p_3, and p_4 formed a third category, and p_6, p_7, and p_8 were allocated to the fourth category (see Figure 2c). A strong effect of the order of stimulus presentation can be observed here. The first presented patterns p_1, p_5, p_2, and p_6 are separately represented on the 4 R-nodes. Subsequently, p_3 and p_7 are allocated to the nearest categories containing p_5 and p_6, respectively. By including p_3 and p_7, the properties of these categories are altered resulting in the respective inclusion of p_4 and p_8 as well. This categorization does not fully reflect the tendency of CALM to cluster similar inputs in 3 equal categories. For example, p_2 and p_4 are members of the same category while their Euclidian distance is larger than the distance between p_2 and p_1 which are members of different categories. In fact, it proves to be unstable. Within additional presentations of the stimulus set the representation of p_2 tends to shift to the R-node representing p_1. Similarly, p_6 tends to shift to the category containing p_5 resulting in the stable categorization shown in Figure 2d where both p_2 and p_6 shifted categories. Figure 2d constitutes the optimal 4 category division of the stimulus space in terms of Euclidian distances between patterns. Due to the stochastic nature of categorization in CALM however, variations in categorization can be observed over different replication trials. In only 40 out of 100 replications, the CALM module reaches the optimal categorization of Figure 2d. Other stable categorizations are shown in Figure 2e and f. These configurations were frequently encountered after a single presentation of the stimuli.

The ability to cluster its inputs forms one of the basic principles governing autonomous categorization in single CALM modules. In the next section, it will be shown that categorization behaviour of such networks can be altered as a result of the cooperative interaction of different modules in multimodule architectures.

3.2. Categorization in Multimodule CALM Networks

It is possible to enhance the CALM so that it always performs an optimal classification of the eight patterns. In this article, however, we want to focus on the role of different architectures in the formation of category structures. We shall, therefore, devote this subsection to the study of a number of small multimodule CALM networks. As a starting point for discussing these networks, we will take the categorization task described in the previous section. It was mentioned that a single module architecture of size 4 only performs an optimal categorization in 40% of 100 replication trials. As above, optimal here refers to a categorization structure that best reflects the cluster structure of the underlying pattern space. In this section, an account will be given of a technique to increase this percentage by adding more CALMs to the network. The main thought behind the construction of the architectures is that a specific categorization or clustering in different modules can facilitate overall performance of the network. The experiments below illustrate how different principles of design can be isolated. A number of multimodule architectures are shown in Figures 4–7. A single thick arrow in the architecture diagrams of these figures indicates full connectivity through modifiable weights between input nodes and R-nodes in the direction indicated by the arrow.

3.2.1. *Principle 1: Structural Compatibility and Neural Assemblies.* Figure 4 shows an architecture containing an input module that is directly, forwardly connected to a CALM of size 4 (module *a*). The task is again to perform an optimal 4-category representation of the stimulus space as shown in Figure 2d. Figure 4 also shows how module *a* has an *indirect* connection with the input through a second CALM of size 2 (module *b*) which is recurrently connected to module *a*. On this task the network produced an optimal categorization in 68 out of 100 replications, which constitutes a considerable improvement over the single module network. The output of module *b* enforces a tendency in module

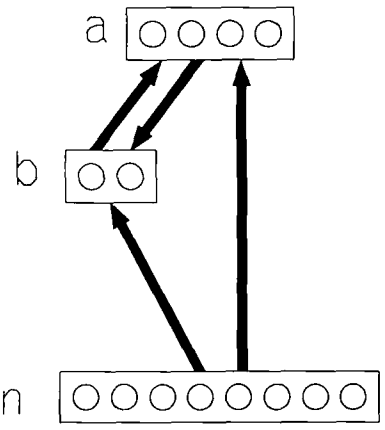

FIGURE 4. Architecture with an input module (in) and CALM modules of size 4 (b) and 2 (a). Only the R-nodes of a module are shown.

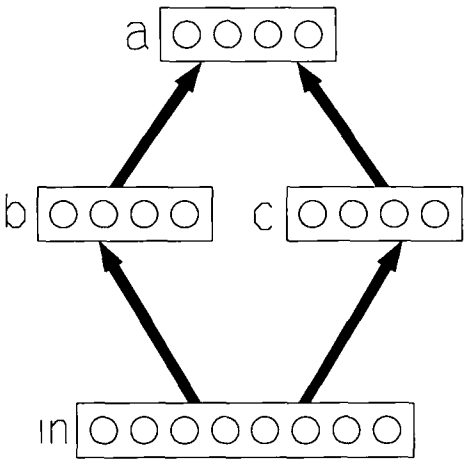

FIGURE 5. Architecture with an input module (in) and three CALM modules of size 4 (a, b, and c). Only the R-nodes are shown.

a to group stimuli of the same main cluster into equal-sized categories, suppressing suboptimal clustering structures (such as illustrated in Figure 2e, f). The coarse categorization performed by module *b* represents the division of the stimulus space shown in Figure 2b, and provides information about stimulus similarities that are relevant to categorization in module *a* and that enhance the clustering process.

Two mutually interacting principles underlie the improved clustering behaviour. First, during the categorization process active R-nodes in module *b* become associated with active R-nodes in module *a* through increased connection weights. Hence, when a stimulus is presented, evidence for a coarse category structure present in module *b* helps rule out irrelevant categories in module *a* by activating relevant categories in module *a*. Second, when subsequent convergences on two or more R-nodes in module *a* coincide with convergence on the same R-node in module *b*, the categories of module *a* become associated with a common category in module *b*. Such common associations encode relations between otherwise independent categories in module *a*. The activation of a category in module *a* will lead to the activation of the associated category in module *b* through the backward connections, which will in turn activate all related categories in module *a*. The simultaneous activation of related categories facilitates shifts of representations among related categories. Because the categorization in module *b* represents the division of the stimulus space in two main clusters, the recurrent weights between module *b* and module *a* encode the main-cluster membership of categories distinguished by module *a*. The backward connections from module *b* induce a topological structure in module *a*. This information is not explicitly encoded in the single module network. The network has a bias for clustering the patterns into two main clusters, but the parameter that determines exactly how this is done is set by exposure to the first few patterns. The multimodule architecture can be said to form a more complete representation of the stimulus space in accordance with the task, which leads to an improvement in overall performance.

Another way of looking at multimodule architectures is in terms of circuits of R-nodes. Each pattern presentation results in the network converging on a set of activated R-nodes that are connected by learning connections. Learning ensures the formation of stable circuits of associations. The idea of such circuits acting as a major determinant of neural coding dates back to Hebb's (1949) idea of neural assemblies. Results such as presented here suggest that the initial architecture of a network is more conducive to the formation of certain assemblies than it is to others. In particular, we would like to argue that the hierarchical and modular structure of an optimal architecture must reflect the overall, hierarchical structure of the relevant input representations. In this way, categorizations are mediated by a set of assemblies inducing a more optimal categorization structure and suppressing the formation of undesirable structures.

An improved categorization in module *a* can also be observed with different stimulus presentation orders (e.g., $p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8$) and when module *b* has size 3. Although overall performance is best when module *b* has two R-nodes, it is evident that a relatively simple coarse spatial processing of stimuli can facilitate a more complex fine-grained processing in a neural network. An interesting way of looking at these models is in terms of multiple size selective channels or separate high and low spatial frequency pathways in the human visual system (e.g., DeYoe & Van Essen, 1988; Living-

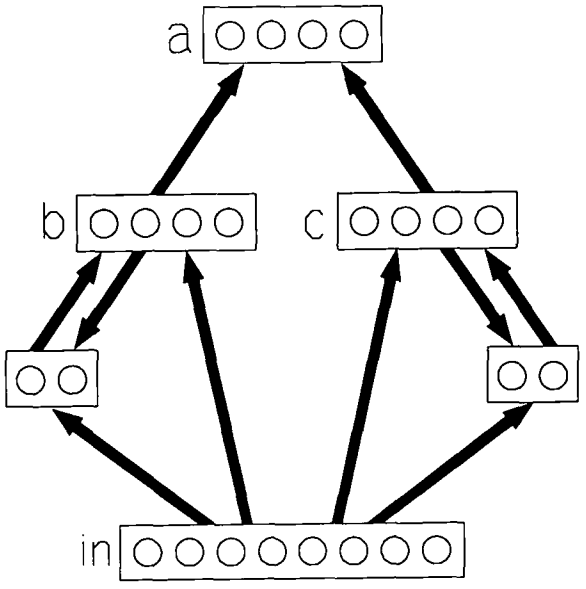

FIGURE 6. Architecture implementing a combination of the two multiple information processing mechanisms.

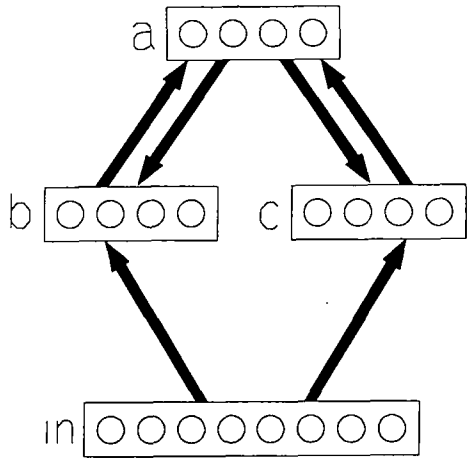

FIGURE 7. Architecture with recurrent weights between module a and module b and c.

stone & Hubel, 1988; Pantle & Sekuler, 1968). The above architecture, although extremely limited, suggests how such subsystems might supplement each other and how they might interact to solve computational difficulties.

3.2.2. *Principle* 2: *Replication of Structure.* A second advantageous mechanism related to temporal aspects of multiple information processing in interactive subsystems can be illustrated using the network shown in Figure 5. The task is again the categorization task described above. The input module of the network is connected to two equivalent processing pathways each consisting of one CALM of size 4 (module b and c), which are both connected to the output CALM (module a) that has to categorize the input stimuli solely on the basis of the output of the other two CALMs. On 100 replications, modules b and c performed like the single module of the architecture of Figure 4, with 39 and 41% optimal categorizations, respectively. Module a on the other hand performed 64% optimal categorizations.

The behaviour of the network can be described as a probability function in which the probability of a performing a correct categorization, $P(a)$, is equal to the probability of either b or c, or both b and c performing a correct categorization ($P(b \vee c)$):

$$P(a) = P(b \vee c) = 1 - P(\neg b \wedge \neg c)$$
$$= 1 - (1 - 0.39)(1 - 0.41) = 0.64.$$

An optimal categorization in one of the modules b or c, will nearly always lead to an optimal categorization in module a. An optimal categorization in both substreams always leads to an optimal categorization in module a. Categorization in a is never optimal when categorization is suboptimal in both b and c. Apparently, both processing pathways complement each other by means of their influencing categorization in a. This can be explained by the observation that a correct categorization in a module is characterized by quicker convergences. Optimal categories cause less competition (i.e., they form deeper attractors) and, thus, take less time to converge. Consequently a substream performing an optimal categorization will be a greater determinant of categorization in c than a substream performing a suboptimal categorization. An optimal categorization agrees better with the internal dynamics of CALM, which is reflected in faster processing.

3.2.3. *Combining Compatibility and Replication.* An even better performance is attained when the categorization in module b and c is improved by adding a module of size 2 to either subsystem (see Figure 6), implementing a combination of both multiple-information processing principles discussed so far. Module b and c both perform 68% optimal categorizations, and module a performs 89% optimal categorizations over 100 replications. This relationship is also described by the above probability function.

The entire architecture shown in Figure 6, can again be duplicated to implement increasingly large architectures that show an increasingly better performance. Apart from its functional implications, this concept of structural replication can be related to phylogenetic issues concerning the development of natural neural architectures. In the expansion of the mammalian neocortex during evolution, for example, replication of existing structure may play an important role (see Killackey, 1990, for a review on this, and related issues).

3.2.4. *Principle* 3: *Recurrence.* A further improvement can be achieved by installing recurrent connections in the architecture of Figure 5, between b and a and between c and a (see Figure 7). Not only categorization in a is improved (90%) but also in b and c (68%). Fast optimal categorization in one substream both facilitates and corrects slower (and initially suboptimal) categorizations in the other substream via the recurrent connections with module a. Furthermore, the probability function described above still applies to the overall categorization behaviour of the modules in the network.

3.2.5. *Interactive Activation.* Modules b and c in architectures 5–7, function as complementary subsystems. Their combined resources result in an improved overall categorization. In addition, when recurrent weights are installed, categorization in a fast pathway can facilitate categorization in a slower pathway. The bidirectional connections carry both bottom-up and top-down information that interactively influences categorization at both levels. The bottom-up connections create an "expectancy" about stimulus categories that

through top-down influences may disambiguate activation that is locally ambiguous (i.e., competition that cannot be resolved quickly). Such a mechanism of interactive activation is also employed by McClelland and Rumelhart (1981) who implement bidirectional connections in a nonlearning network to simulate a series of experiments in letter detection. CALM is fully compatible with this type of modelling. In fact, Murre et al. (1992) describe a learning CALM version of the interactive-activation model of letter perception. The ideas of interactive activation can be traced back to Rumelhart (1977) and to Grossberg (1978). It is an important psychological concept that can be applied to the understanding of a wide range of phenomena. Most of the references dealing with competition, cited above, are also concerned with interactive-activation processing. The dynamics of the described CALM architectures suggest specific ways in which different modules can cooperate. The above findings, for instance, agree well with Todd (1985), who remarks about the visual system that: ". . . objects and events in a natural environment can be multiply specified by many different sources of information, each of which is detected by a specialized processing module with its own individual limitations. In any given situation, we would expect to obtain erroneous outputs from some of these modules because of inappropriate viewing conditions, but it would be most unlikely for two or more of them to fail in exactly the same way. Thus, if different modules could be designed to excite one another when their outputs are compatible and to inhibit one another when their outputs are incompatible then the inappropriate modules would be dynamically suppressed, and the system would eventually converge on a correct interpretation of the available information (Todd, 1985: 708)." With the models analyzed above the multiple specification of information in separate modules even leads to a correction, instead of the mere inhibition of processing, in incompatible modules.

To conclude this section, it has become evident that the CALM can be effectively used as a building block in modular network architectures. Moreover, such models seem to be compatible with notions about neural structures and information processing in natural neural systems. The approach outlined here combines the concept of modularity with Hebb's (1949) idea of neural assemblies. It is important to note that the modular architecture of these networks is *not* the result of a learning process. Instead, what is learned depends on the initial architecture. It was, for instance, shown how a small module enforces a coarse categorization, which is conducive to learning favorable between-category relations. The small categorization task used in this section made it possible to design by hand simple modular architectures, based on three principles of design that underly all interactive activation networks. In the remainder of this study, network design methods will be applied to the more complex visual task of handwritten digit categorization.

4. EVOLUTION OF MODULAR NETWORK ARCHITECTURES

This study focuses on modularity as a basic design principle to implement effective and neuroanatomically plausible constraints on interactive neural networks. Our experience with connectionist systems for learning complex tasks so far indicates that for many tasks it is difficult to find an effective functional architecture and to select the appropriate values for the system parameters. Although above we have indicated how it is possible to isolate several principles of design, we know of no comprehensive analytical solution to the problem of relating architecture to function. This is a subject well worthy of extensive analytical investigation. Here, however, we will focus on a method that makes very few assumptions about the functions to be performed by the architecture. Just as learning in neural networks is an autonomous self-organizing method for finding suitable weight values, we will use an autonomous self-organizing procedure to find optimal network architectures and parameter values.

In biology the search for efficient neural architectures is governed by Darwinian evolution. Nearly all initial neuroanatomical structure (or all structure for that matter) of an organism is the result of this very powerful and creative search process. We propose to extend the biological metaphor in connectionist modelling by using genetic algorithms to find efficient modular architectures. Genetic algorithms (Holland, 1975) are autonomous search algorithms modelled after biological genetic search or evolution. They have proven to be very effective in finding solutions to complex problems with many, mutually dependent constraints. The design of effective neural network architectures is such a problem. In this section, we will first briefly review some basic principles of biological adaption and genetic algorithms. This is followed by an experiment in the artificial evolution of CALM networks.

4.1. Biological Adaptation

In the general process of biological adaptation we may distinguish four principal levels. Each of these levels is concerned with specific environmental circumstances and operates on a different time scale. *Evolution* can be considered as the first and highest level of adaptation. It involves the adaptation of a species to global ecological and environmental conditions. This adaptation is a relatively slow process that operates over millennia, although the speed of genetic adaptation may differ widely for individual species (e.g., viruses can change their genetic properties within days).

Ontogenesis, or the development of an individual

organism, is the second lower level of adaptation, operating on a time scale measured in years, months, or weeks. Ontogenesis in the nervous system involves the development of specific neural structures and the degeneration of other neural structures. Massive regressive events of neuronal connectivity in the vertebrate nervous system can be seen as part of the development and maturation of neural functions (e.g., Changeux & Danchin, 1976; Cowan, Fawcett, O'Leary, & Stanfield, 1984; Rakic, Bourgeois, Eckenhoff, Zecevic, & Goldman-Rakic, 1985).

Learning forms the third level of adaptation of an organism to its specific environment. Learning can be seen as a fine tuning of the neural structures that were phylogenetically and ontogenetically established. Learning processes operate at an even smaller time scale of minutes or seconds.

At the fourth level, we have the dynamics of *neural activation* that operate within milliseconds and that form a fast and specific adaptation mechanism that enables an organism to act on the demands of the moment.

These four processes, although tentatively described as independent levels of adaptation, are strongly interrelated. In the previous section, for example, we described how learning could be guided by the initial network architecture, which in organisms is the result of evolution. In this way, it can be argued that *evolution guides learning.* Although, it is pointed out by Chalmers (1990) that in a relatively static environment there might be little need for learning systems to evolve, it can also be argued that the nature of the evolutionary process itself is altered greatly when evolving organisms acquire an increased capacity to learn, to the extent that we can say that *learning guides evolution* (also see Belew, 1989; Hinton & Nowlan, 1987). There is thus a strong mutual dependency between the different levels. Recent theorizing in the philosophy of biology also stresses the interrelatedness of local and global levels in evolution and adaptation. Depew and Weber (1989; also see Weber, Depew, Dyke, Salthe, Schneider, Ulanowicz, & Wicken, 1989), for instance, have presented a well-argued proposal to integrate the global, ecological perspective and the local, developmental perspective in neo-Darwinist evolutionary theory. Their arguments are of a general and philosophical nature, and the theory is not applied to specific instances. Recent simulation studies of artificial ecosystems provide good supplementary illustrations of their viewpoint. Ackley and Littman (1990; more fully described in Ackley & Littman, in prep), for example, studied evolution in an artificial world. They were specifically concerned with the interrelation between learning and evolution. An important conclusion was that learning may provide a decisive evolutionary advantage. Genes enhancing learning in organisms have a higher chance of survival than less adaptive variants. These and other relations between learning and evolution may, ultimately, lead us to a combined genetic–connectionist approach in the modelling of adaptive systems.

4.2. Genetic Algorithms

Genetic algorithms are defined by Goldberg (1989, p. 1) as ". . . search algorithms based on the mechanics of natural selection and natural genetics." The process of evolution is used as a "real-world model" that serves as a source of ideas for solving practical and theoretical problems in modelling and optimization. Just as neural networks use a brain metaphor, genetic algorithms use an evolution metaphor. The general use of biological metaphors in modelling and computing has strongly increased in the past decade. The various approaches based on these metaphors can be jointly referred to as *biocomputing* (Valdès, 1991). In addition to neural networks and genetic algorithms, it also includes reiterated affine transformations (Barnsely, 1988), fractal systems (Mandelbrot, 1982), fuzzy logic (Kandel & Lee, 1979; Zadeh, 1987), chaos theory (Gleick, 1987), simulated annealing (Kirkpatrick, Gelatt, & Vecchi, 1983), and the study of complex systems (Prigogine & Stengers, 1984). The success of nature in solving many problems that are nowadays recognized as very difficult for the traditional approaches, have led researchers into studying the biological example. In various abstractions and formalizations, biological systems have been theoretically proven to provide robust solutions to these hard problems.

Genetic algorithms were introduced by John Holland (1975). For a thorough and up-to-date overview of genetic algorithms the reader is referred to the excellent introduction by Goldberg (1989), that includes not only all the basics of genetic algorithms but also a description of computer software (in Pascal), and a seemingly exhaustive overview of applications of genetic algorithms to a wide range of problems. Some illustrative examples of genetic algorithm applications are: the optimization of VLSI circuits (Davis & Smith, 1985), optimizing the connections in communication networks (Davis & Coombs, 1987), and the selection and creation of feature detectors for pattern recognition systems (Chang, Lippmann, & Tong, 1990; Englander, 1985). In what follows, some of the basics of genetic algorithms will be briefly treated.

Goldberg (1989, p. 7) mentions four differences between genetic algorithms and other search methods.

1. Genetic algorithms work with a *coded* parameter set.
2. They search from a *population* of points in a solution space, rather than from a single point.
3. They only use directly available information provided through a *fitness function.*
4. They rely on *probabilistic transition rules* instead of deterministic rules.

The relation between the genetic code and the developing organism may be strongly nonlinear: small changes in a gene may, but need not, result in major changes in the organism (e.g., Dawkins, 1986). The basic mechanism of natural selection, however, is straightforward. An organism in a population has a certain chance to live and reproduce, which is usually referred to as its *fitness*. Fit organisms are more likely to live and reproduce than nonfit organisms. Or from a slightly different perspective: *fit genes live, others die* (also see Dawkins, 1976, for a convincing defense of the "gene perspective").

As with neural networks, the terms used for genetic algorithms differ from their biological counterparts. Genetic algorithms work with strings of (usually) binary features. Each string encodes a possible solution to a problem. Starting from an initial population that may consist of random strings, the search for optimal solutions proceeds through the general process of *reproduction*. New strings (i.e., coded solutions) are created and replace less fit solutions in the present population. From generation to generation, this leads to a higher overall fitness of the population and to better solutions. Reproduction is mediated by a number of genetic operators. The most commonly used genetic operators are *selection, crossover,* and *mutation*.

The *selection* operator selects strings from the population for reproduction. In general, strings are randomly selected, with high-fitness strings having a higher chance of being selected. For example, the probability of a string to be selected can be taken proportional to its absolute fitness value (e.g., Goldberg, 1989). Another possibility is *rank-based selection* (Whitley, 1989), in which the selection probability is defined as a linear function of the *rank* of a string in the population according to its fitness. With all simulations in this study, the *static population model* (Whitley, 1989) will be used for selection and replacement of solutions. Apart from rank-based selection this model uses *one-at-a-time selection and replacement*. A new solution is inserted at the appropriate position (rank) and "pushes the worst solution off the table." The best solutions always stay within the population. The best fitness value thus increases monotonically.

With *crossover,* two selected strings are combined to produce a new solution. Usually two *crossover points* are randomly chosen at which the two parent strings are cut. After parts of both strings have been interchanged they are "glued" together to produce two new strings. It can be shown that this principle of structured information exchange leads to an exponential increase of highly fit pieces of genetic information. Such pieces of short defining length are called *building blocks*. They can be seen as coding partial solutions to the problem at hand. These partial solutions are combined by the genetic algorithm to produce good overall solutions. The notion that the combination of fit building blocks leads to better overall solutions is called the *building block hypothesis* (Goldberg, 1989).

Randomness may enter into the genetic algorithm at various points. The initial population is usually randomly generated, selection has a strong random component, crossover is applied with a specific probability, and the cutpoints are randomly chosen. A final genetic operator is called mutation, which adds yet another random component to the genetic algorithm. Mutation randomly flips single bits from 0 to 1 or from 1 to 0. Mutation may improve the performance of the genetic algorithm by occasionally suggesting a new partial solution not present in the population, and by protecting it against the accidental, irrecoverable loss of valuable information due for example to unfavorable crossovers. The mutation probability of a bit is usually chosen to be low, in order to minimize interference with the main genetic operations.

Recently, researchers have started to combine genetic algorithms with neural networks to form more powerful adaptive systems (e.g., Dodd, 1990; East & Macfarlane, 1990; Macfarlane, 1992; Maricic & Nikolov, 1990; Miller, Todd, & Hedge, 1989; Whitley & Bogart, 1990; Whitley & Hanson, 1989; Whitley & Starkweather, 1990). Others have taken a more direct approach, where not only the topology and learning parameters are specified by the genetic algorithm, but also all weight values and thresholds (e.g., De Garis, 1990a,b; Harp, Samad, & Guha, 1989). A much promising and biologically more plausible method is the evolution of grammar based encodings of neural network structures (e.g. Boers, Kuiper, Happel, & Sprinkhuizen-Kuyper, 1993; Gruau, 1991, 1992; Kitano, 1990; Merrill & Port, 1991). In many of these studies a comparison is also made with traditional search methods. From these comparisons it appears that the combination of neural networks and genetic algorithms holds promise for practical applications. Genetic algorithms are relatively insensitive to local minima and quick at finding approximate solutions, but they are typically slow in fine tuning these solutions. Learning in neural networks, when viewed as a search method, is often sensitive to local minima and generally better at fine-tuning. Constraints on network architecture form a major determinant of the learning capacity of a neural network. Specification of this architecture by a genetic algorithm, therefore, seems to be a good way to combine learning and genetic search. In this approach, learning imparts a finer structure on a neural network coarsely lined-out by a genetic algorithm. Genetic algorithms are used to move the search to an appropriate region in the solution space. Learning then executes a more local search to achieve an optimal performance. In the light of the forgoing simulations with small CALM networks, a promising approach to solving complex problems with learning neural networks is to have a genetic algorithm specify a *modular* initial architecture.

4.3. Exploring Modular Architectures

4.3.1. *Categorization of Handwritten Digits.* For the experiments in the remainder of this section, categorization of unconstrained, handwritten digits was chosen as a case study. This problem was selected because it deals with natural, visual objects in a real two-dimensional space and because the mapping from image space to category space has both considerable regularity and complexity. The digits 0–9 were collected in different handwritings (see Happel, 1990; Happel, Phaf, Murre, & Wolters, 1990, for details). In two experiments the objective was to have a genetic algorithm find optimal, modular network architectures that learn to classify all handwritten digits using only a few learning trials. Network architectures generated by the genetic algorithm were trained with just a single exemplar of every digit category. In the first experiment the fitness of network architectures was measured by their performance on *recognition* of the previously trained patterns, while in the second experiment *generalization* performance on digits not encountered before provided the fitness values.

Both the modular structure of a network and a *global* set of parameters associated with the activation and learning dynamics of CALM were defined by the genetic algorithm. This set of 17 parameters applied to all modules in a network architecture. The generated networks consisted of an input module, zero or more intermediate CALMs, and a CALM from which the classification was derived (the output CALM). A network was trained by presenting a set of input patterns five times. Each presentation of a single pattern lasted for 50 network cycles. Between stimulus presentations all activations in the network were reset. A training set consisted of 10 handwritten digits that were randomly selected from a pool of 200 digits. Training was unsupervised. To facilitate learning of the training set, however, a category was designated by presenting a *symbolic cue* one cycle before and during the presentation of a digit. These cues consisted of 10 orthogonal binary input patterns that were fed into a network through 10 additional nodes in the input module. Cueing categories facilitated unsupervised categorization and learning because it enlarges the Euclidian distance of patterns belonging to different categories. Note, however, because cues are orthogonal they do not change the structure or cluster properties of the stimulus space, nor do they provide any information about similarity relations between different categories. These have to be derived by a network architecture solely from handwritten digit input. After training, networks were tested on recognition of the 10 trained patterns or generalization performance on the 190 untrained patterns, both without the benefit of category cueing. In order to perform a correct categorization in the test phase, a network had to represent a test pattern on the R-node in the output module that became associated with the corresponding digit category during the training phase. Learning was prohibited in the test phase.

It should, perhaps, be pointed out that the primary focus here is on exploring modular architectures and not on the solution of a particular real-world problem (i.e., digit recognition). The absolute performance level of the networks described here is not as high as results obtained with neural network approaches that are exclusively dedicated to digit recognition (for an overview of neural networks performance on these tasks, see, for instance, Guyon, Poujaud, Personnaz, & Dreyfus, 1989). The networks described below were fed with a very coarse resolution input consisting of 25 continuous activation values. Preprocessing was limited to centering the input patterns and a normalization of their size. Also, the size of the task and of the networks was kept small, because the use of genetic algorithms necessarily entails the computationally very demanding exploration of many different network structures. For a detailed description of the methods used the reader is referred to the Appendix.

In the following experiments, a genetic search is conducted for optimal network solutions to the handwritten digit recognition problem. The fitness of a solution, which determines its rank in the population, is obtained by training and testing the network it encodes. Training and testing proceed as described above. Either the percentage correct recognition or the percentage generalization constituted the fitness value. Because, as mentioned, we use a static population model for selection and replacement (Whitley, 1989), the mean fitness of a population simply increased monotonically during evolution until it asymptotically reached its maximum. Not until that point genetic diversity in the population reached a minimum and an optimal solution had been found.

4.3.2. *Experiment* 1: *Recognition.* For the first experiment, the recognition performance of an architecture on the 10 training digits was used as the fitness criterion. After 30,000 evaluated recombinations the search converged. Nearly all solutions present in the population encoded single module architectures directly connecting the input module to the output module, without hidden modules. The best networks in the population were subjected to a test involving 500 replications of the above described train and test procedure. For each replication a different training set was selected from the pool of 200 digits. Network performance was tested on recognition of the training set, on generalization to the 190 untrained digits, and on generalization performance to 100 completely new digits, that were not part of the pool used for genetic optimization. The best results on both recognition and generalization performances were obtained with a single module architecture that scored a mean of 98.4% correct recognition. Gen-

eralization scores, which did not contribute to the fitness criterion, were 52.3% on the untrained digits and 49.8% correct generalization on the new digits.

With respect to the performance of a standard CALM, single module architecture, the recognition and generalization performance strongly improved. This must be entirely attributed to the optimized parameter set installed by the genetic algorithm. The most prominent changes in the dynamics of CALM as a result of the new parameters, involved a reduced noise level along with an increased competition level and an increased learning rate. In the light of the improvements obtained with multimodule architectures in the previous section, however, it comes somewhat as a surprise that a single-module architecture in this case performs best. Apparently, optimal rule extraction from a limited set of examples can be obtained with only a single layer of learning weights between input module and output module. Although in the early stages of the evolutionary process the algorithm installed some multimodule structures, these were later eliminated during convergence of the search toward an optimal solution. As mentioned earlier, learning in largely unstructured networks may often result in good specific solutions that incorporate rules that effectively describe the regularities and characteristics of the training set. Such specific solutions, however, usually lack generality. In order to find good general solutions, knowledge about the total task domain needs to be incorporated in the network. This was, for example, illustrated by the CALM networks of the previous chapter, that needed specific modular architectures to reduce network entropy in order to guide learning toward a solution. This notion is also reinforced by the following, second experiment.

4.3.3. *Experiment 2: Evolving Generalizing Networks.* This is a replication of Experiment 1. The only difference is the fitness criterion, which now exclusively measures generalization performance. The networks were first trained on the 10 digits of the training set. Then they were tested with the untrained digits. The fitness score was based on the percentage correct categorization achieved on the latter set.

During the genetic search, the same type of multimodule solutions emerged as early on in the previous simulation, only this time they were not eliminated and within 30,000 recombinations completely dominated the population. A typical example of the architectures produced by the genetic algorithm is shown in Figure 8. The network uses 10 nodes distributed over 4 hidden modules, one of size 4, and three of size 2. The hidden modules are intricately connected to the input module and output module. The evolved parameter settings induce a high learning rate, a high level of competition between the R-nodes of a module, and a much reduced noise level, which is basically limited to one short initial burst of random activation, necessary to break the symmetry.

When subjected to the same 500 replicated tests used to evaluate the performance of the networks in experiment 1, this architecture reached a mean recognition score of 99.1% on the 10 trained digits, 58.0% correct generalization on the 190 untrained digits, and 54.4% correct generalization on the 100 new digits. In a paired t-test the improvement in generalization scores over the best performing network of experiment 1, proved to be highly significant ($t = 40.473$ and $t = 30.280$, respectively). Even the small difference in recognition performance of 0.7% was significant well within the 0.001 confidence interval ($t = 8.246$).

In the architecture of Figure 8, a direct connection from the input module to the output CALM is supplemented by four hidden CALMs. This structure can be seen as a complex subcategorization circuit processing information at different, coarser levels. Also, a principle of replication of structure can be identified in the two hidden modules of size 2 that receive input from the module of size 4. Both modules are, however, differently connected to the rest of the network. Recurrent circuits are implemented by various feedback connections. The CALMs in the network have developed a relatively high competition level between R-nodes, resulting from a strong inhibitory influence of the internal veto nodes.

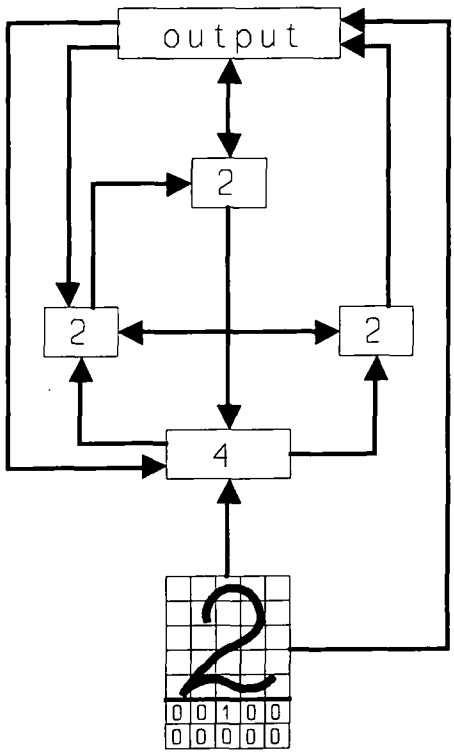

FIGURE 8. CALM network architecture. Arrows denote intermodular, learning connections. Numbers indicate the size of the CALM modules.

This inhibition results in oscillations of the R-node activations of a module. These oscillations introduce a strong temporal aspect of information processing in the network, characterized by the repeated, asynchronous activation and "resetting" of the modules. Resetting here refers to inhibition of all R-nodes, so that most of them reach a zero activation level. Together with the recurrent loops present in the architecture, this implements a mechanism that seems to accumulate evidence over subsequent oscillations for a specific classification. This behavior may be compared to the model by Ambros-Ingerson, Granger, and Lynch (1990) of the olfactory paleocortex. Successive "sniffs" narrow down the representation for a particular pattern.

5. DISCUSSION

The main conclusion that can be drawn from the above two experiments is that an initial modular architecture can induce a system to better generalize its learned behaviour to instances never encountered before. In both simulations the networks were exposed to equally limited training data. Recognition scores on these patterns differed only slightly for both networks. Yet, generalization performance of the architecture of experiment 2 was much better. This result was obtained consistently in three full replications of Experiments 1 and 2, with different random initial populations, and various other simulations with larger training sets as well as fixed training and test sets (Happel & Murre, 1992). All simulations that used generalization to measure fitness arrived at modular configurations that were very similar, indicating that they form indeed functionally advantageous architectures that capture important regularities present in the input domain.

The structure of these networks is reminiscent of the broad division of the primate visual system into two principal pathways (e.g., Livingstone & Hubel, 1988). The magnocellular pathway processes visual input in a coarse manner and has a fast response, the parvocellular pathway carries out a much more detailed analysis and has a much slower operating characteristic. Furthermore, the modular structure of the generalization network of experiment 2 bears a resemblance to the global modular organization of the primate visual system (e.g., see Zeki & Shipp, 1988). It may, therefore, illustrate some important modular principles of interactive information processing, characteristic of both artificial and natural neural systems.

In addition, some of the networks mentioned demonstrate other properties that are characteristic of natural neural systems, such as the ability to learn patterns in a sequential manner without much interference with existing representations. This problem has plagued neural networks based on both error-correction learning and unconstrained architectures, such as back propagation (McCloskey & Cohen, 1989; Ratcliff, 1990). It has been shown that most of this tendency to overwrite existing representations, thereby causing "catastrophic interference," can be attributed to overlapping hidden-layer representations (French, 1991; Murre, 1992a). In fact, it can be argued that all methods introduced to diminish sequential interference in back propagation function by reducing hidden-layer overlap (Murre, 1992b). In particular, the introduction of modularity as outlined above or "semidistributed" representations (French, 1991) can strongly alleviate interference.

6. CONCLUSIONS

The initial architecture of a neural network, whether artificial or natural, may guide learning. The above two experiments in genetic design suggest that the evolutionary directives encoded in the structure of the brain may extend beyond merely an increased ability to learn stimuli necessary for survival. We propose that the initial architecture is not only important for rapid learning, but that it also induces the system to generalize its learned behaviour to instances not previously encountered. Generalization of learning may well be a principal function of much of the initial structure of the brain. This would explain why for many vital learning tasks only a minimal exposure to relevant stimuli is necessary. Evolution coarsely programs the brain to function in specific task domains. Learning completes these neural programs by fine-tuning the connections and dynamics. The combination of an initial architecture produced by evolution and experience-based additional fine-tuning prepares the organism to function in an entire domain, rather than just the limited part of the environment to which it was exposed. If this is indeed an important underlying principle, we must conclude that the hidden structure of the brain may capture many more regularities of the world around us than we have expected so far. Further analysis and simulation with modular neural networks and genetic search can help us understand how these regularities are embedded in the brain.

REFERENCES

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, **9**, 147–169.

Ackley, D. H., & Littman, M. S. (1990). Learning from natural selection in an artificial environment. *Proceedings of the International Joint Conference on Neural Networks, Washington DC*, 1, Hillsdale, NJ: Lawrence Erlbaum, 189–193.

Ackley, D. H., & Littman, M. S. (in preparation). *Evolutionary reinforcement learning.*

Allport, D. A. (1980). Patterns and actions. In G. L. Claxton (Ed.), *New directions in cognitive psychology.* London: Routledge & Kegan Paul.

Ambros-Ingerson, J., Granger, R., & Lynch, G. (1990). Simulation of paleocortex performs hierarchical clustering. *Science*, **247**, 1344–1348.

Anderson, J. R., & Bower, G. H. (1974). *Human associative memory.* Washington DC: Hemisphere Publishing Corporation.

Barna, G., & Kaski, K. (1990). Choosing optimal network structure. In B. Widrow & B. Angeniol (Eds.), *Proceedings of the International Neural Network Conference, INNC-90-Paris.* Dordrecht: Kluwer, 890–893.

Barnsley, M. J. (1988). *Fractals everywhere.* San Diego CA: Academic Press.

Baum, E. B., & Haussler, D. (1989). What size net gives valid generalization? *Neural Computation,* **1,** 151–160.

Belew, R. K. (1989). When both individuals and populations search: Adding simple learning to genetic algorithms. *Proceedings of the International Conference on Genetic Algorithms and their Applications,* 34–41.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. K. (1989). Learnability and the Vapnik–Chervonenskis dimension. *Journal of the ACM,* **36,** 929–965.

Boers, E. J. W., Kuiper, H., Happel, B. L. M., & Sprinkhuizen–Kuyper, I. G. (1993). Designing modular artificial neural networks. *Proceedings of Computing Science in the Netherlands, CNS'93.* Amsterdam: Stichting Mathematisch Centrum, 87–96.

Bridgeman, B. (1971). Metacontrast and lateral inhibition. *Psychological Review,* **78,** 528–539.

Brodmann, K. (1909). *Vergleichende Lokalisationslehre der Grosshirnrinde in ihren Prinzipien dargestellt auf Grund des Zellenbaues.* Leipzig: Barth.

Bülthoff, H. H., & Mallot, H. A. (1987). Interaction of different modules in depth perception. *Proceedings of the First International Conference on Computer Vision.* London: IEEE Computer Society, 295–305.

Carpenter, G. A., & Grossberg, S. (1987). Neural dynamics of category learning and recognition: Attention, memory consolidation, and amnesia. In J. Davis, R. Newburgh, & E. Wegman (Eds.), *Brain structure, learning, and memory.* AAAS Symposium Series.

Carpenter, G. A., & Grossberg, S. (1988). The ART of adaptive pattern recognition by a self-organizing neural network. *Computer,* **21,** 77–88.

Cavanagh, P. (1987). Reconstructing the third dimension: Interactions between colour, texture, motion, binocular disparity and shape. *Computer Vision, Graphics, and Image Processing,* **37,** 171–195.

Chalmers, D. J. (1990). The evolution of learning: An experiment in genetic connectionism. *Proceedings of the* 1990 *Connectionist Models Summer School.* San Mateo, CA: Morgan Kaufmann.

Chang, E. I., Lippmann, R. P., & Tong, D. W. (1990). Using genetic algorithms to select and create features for pattern classification. *Proceedings of the International Neural Network Conference, San Diego,* **3,** 747–752.

Changeux, J., & Danchin, A. (1976). Selective stabilisation of developing synapses as a mechanism for the specification of neural networks. *Nature,* **264,** 705–712.

Cornsweet, T. N. (1970). *Visual perception.* New York: Academic Press.

Cowan, W. M., Fawcett, J. W., O'Leary, D. D. M., & Stanfield, B. B. (1984). Regressive events in neurogenesis. *Science,* **225,** 1258–1265.

Creutzfeldt, O. D. (1977). Generality of the functional structure of the neocortex. *Naturwissenschaften,* **64,** 507–517.

Damasio, A. R., Damasio, H., & Van Hoesen, G. W. (1982). Prosopagnosia: Anatomic basis and behavioral mechanisms. *Neurology,* **32,** 331–341.

Davis, L., & Coombs, S. (1987). Optimizing network link sizes with genetic algorithms. In M. Elzas, T. Oren, & B. P. Zeigler (Eds.), *Modelling and simulation methodology: Knowledge systems paradigms.* Amsterdam: North–Holland.

Davis, L., & Smith, D. (1985). *Adaptive design for layout synthesis.* Texas Instruments internal report. Dallas: Texas Instruments.

Dawkins, R. (1976). *The selfish gene.* Oxford: Oxford University Press.

Dawkins, R. (1986). *The blind watchmaker.* New York: Norton.

De Garis, H. (1990a). Brain building with GenNets. *Proceedings of the International Neural Network Conference, Paris,* 2. Dordrecht: Kluwer, 1036–1039.

De Garis, H. (1990b). Genetic programming: Modular neural evolution for Darwin machines. *Proceedings of the International Joint Conference on Neural Networks, Washington DC,* 1. Hillsdale, NJ: Lawrence Erlbaum, 194–197.

Depew, D. J., & Weber, B. H. (1989). The evolution of the Darwinian research tradition. *Systems Research,* **6,** 255–263.

DeYoe, E. A., & Van Essen, D. C. (1988). Concurrent processing streams in monkey visual cortex. *Trends in Neurosciences,* **11,** 219–226.

Dodd, N. (1990). Optimisation of network structure using genetic techniques. *Proceedings of the International Neural Network Conference, Paris,* 2. Dordrecht: Kluwer, 693–696.

East, I. R., & Macfarlane, D. (1990). An investigation of several parallel genetic algorithms. In Turner (Ed.), *Proceedings of the 12th Technical Meeting of the Occam User Group,* IOS.

Eccles, J. C. (1981). *Neuroscience,* **6,** 1839–1855.

Englander, A. C. (1985). Machine learning of visual recognition using genetic algorithms. *Proceedings of the International Conference on Genetic Algorithms and their Applications,* 197–201.

Frean, M. (1990). The upstart algorithm: A method for constructing and training feedforward neural networks. *Neural Computation,* **2,** 198–209.

French, R. M. (1991). Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks. *Proceedings of the 13th Annual Cognitive Science Society Conference,* Hillsdale, NJ: Lawrence Erlbaum, 173–178.

Fritzke, B. (1991). Let it grow—self-organizing feature maps with problem dependent cell structure. *Proceedings of the* 1991 *International Conference on Artificial Neural Networks, Volume* 1, Amsterdam: North–Holland, 403–408.

Fukushima, K. (1975). Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics,* **20,** 121–136.

Fukushima, K. (1988). Neocognitron, a hierarchical neural network capable of visual pattern recognition. *Neural Networks,* **1,** 119–130.

Funahashi, K-I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks,* **2,** 183–192.

Gazzaniga, M. S. (1989). Organization of the human brain. *Science,* **245,** 947–952.

Gleick, J. (1987). *Chaos: Making a new science.* New York: Viking.

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

Graf, P., & Mandler, G. (1984). Activation makes words more accessible, but not necessarily more retrievable. *Journal of Verbal Learning and Verbal Behaviour,* **23,** 553–568.

Grossberg, S. (1976a). Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors. *Biological Cybernetics,* **23,** 121–134.

Grossberg, S. (1976b). Adaptive pattern classification and universal recoding: II. Feedback, expectation, olfaction, and illusions. *Biological Cybernetics,* **23,** 187–202.

Grossberg, S. (1978). A theory of human memory: Self-organization and performance of sensory motor codes, maps and plans. In R. Rosen & F. Snell (Eds.), *Progress in Theoretical Biology, Volume* 5. New York: Academic Press, 233–374.

Grossberg, S. (1982). *Studies of mind and brain.* Boston: Reidel Press.

Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science,* **11,** 23–63.

Gruau, F. (1991). *Synthese de reseaux de neurones par algorithme genetique.* Masters thesis, Ecole Normale Superieure de Lyon.

Gruau, F. (1992). Genetic synthesis of Boolean neural networks with a cell rewriting developmental process. In D. Whitley & J. D. Schaffer (Eds.), *Combinations of Genetic Algorithms and Neural Networks,* Washington, DC: IEEE Computer Society Press.

Guyon, I., Poujaud, I., Personnaz, L., & Dreyfus, G. (1989). Comparing different network architectures for classifying handwritten digits. *Proceedings of the International Joint Conference on Neural Networks*, **1**, 127–132.

Happel, B. L. M. (1990). *Pattern categorization in multi-module CALM networks: Recognition of handwritten digits.* Unpublished Master's Thesis, Department of Psychology, Leiden University.

Happel, B. L. M., & Murre, J. M. J. (1992). Designing modular network architectures using a genetic algorithm. In I. Aleksander, & J. Taylor (Eds.), *Artificial neural networks, 2, Proceedings of the* 1992 *International Conference on Artificial Neural Networks, Volume* 2, Amsterdam: North-Holland, 1215–1218.

Happel, B. L. M., Phaf, R. H., Murre, J. M. J., & Wolters, G. (1990). Categorization in multi-module CALM networks: Recognition of handwritten digits. *Proceedings of the International Neural Network Conference, INNC-90-Paris,* Dordrecht: Kluwer, 51.

Harp, S. A., Samad, T., & Guha, A. (1989). Toward the genetic synthesis of neural networks. *Proceedings of the Third International Conference on Genetic Algorithms and their applications.* Massachusetts Institute of Technology, 360–369.

Haussler, D. (1990). *Probably approximately correct learning.* Technical report UCSC-CRL-90-16, Computer Research Laboratory, University of California at Santa Cruz, Santa Cruz, CA.

Hebb, D. O. (1949). *The organization of behaviour.* New York: Wiley.

Hebb, D. O. (1955). Drives and the conceptual nervous system. *Psychological Review,* **62,** 243–254.

Hilz, R., & Rentschler, I. (1989). Segregation of color and form. *Naturwissenschaften,* **76,** 479–480.

Hinton, G. E. C., & Nowlan, S. J. (1987). How learning can guide evolution. *Complex Systems,* **1,** 495–502.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor, MI: University of Michigan Press.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences USA,* **79,** 2554–2558.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks,* **2,** 359–366.

Hornik, K., Stinchcombe, M., & White, H. (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks,* **3,** 551–560.

Hubel, D. H., & Wiesel, T. N. (1959). Receptive fields of single neurons in the cat striate cortex. *Journal of Physiology,* **148,** 574–591.

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology,* **160,** 106–154.

Hubel, D. H., & Wiesel, T. N. (1965). Receptive fields and functional architecture in two nonstriate visual area (18 and 19) of the cat. *Journal of Neurophysiology,* **28,** 229–289.

Kandel, A., & Lee, S. C. (1979). *Fuzzy switching and automata: Theory and applications.* New York: Crane Russak.

Kandel, E. R., & Schwartz, J. H. (1985). *Principles of neural science.* New York: Elsevier.

Killackey, H. P. (1990). Neocortical expansion: An attempt toward relating phylogeny and ontogeny. *Journal of Cognitive Neuroscience,* **2,** 1–17.

Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science,* **220,** 671–680.

Kitano, H. (1990). Designing neural network using genetic algorithm with graph generation system. *Complex Systems,* **4,** 461–476.

Kosslyn, S. M., Flynn, R. A., Amsterdam, J. B., & Wang, G. (1990). Components of high-level vision: A cognitive neuroscience analysis and accounts of neurological syndromes. *Cognition,* **34,** 203–277.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1990). Handwritten digit recognition with back-propagation network. In D. Touretzky (Ed.), *Information processing systems, Volume* 2, San Mateo, CA: Morgan Kaufmann.

Livingstone, M., & Hubel, D. (1988). Segregation of form, color, movement, and depth: Anatomy, physiology, and perception. *Science,* **240,** 740–749.

Macfarlane, D. (1992). *A practical investigation of parallel genetic algorithms and their applications to the structuring of artificial neural networks.* Ph.D. Thesis, University of Buckingham, Buckingham UK.

Mandelbrot, B. B. (1982). *The fractal geometry of nature.* San Francisco CA: Freeman.

Mandler, G. (1980). Recognizing: The judgment of previous occurrence. *Psychological Review,* **87,** 252–271.

Maricic, B., & Nikolov, Z. (1990). GENNET—Systems for computer aided neural network design using genetic algorithms. *Proceedings of the International Joint Conference on Neural Networks, Washington DC,* 1. Hillsdale, NJ: Lawrence Erlbaum, 102–105.

Marshall, J. C., & Newcombe, F. (1973). Patterns of paralexia: A psycholinguistic approach. *Journal of Psycholinguistic Research,* **2,** 175–199.

McClelland, J. L., & Rumelhart, D. E. (1981). An interactive activation model of context effects in letter perception: Part 1: An account of basic findings. *Psychological Review,* **88,** 375–407.

McCloskey, M., & Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In G. H. Bower (Ed.), *The psychology of learning and motivation.* New York: Academic Press.

Merrill, J. W. L., & Port, R. F. (1991). Fractally configured neural networks. *Neural Networks,* **4,** 53–60.

Miller, G. F., Todd, P. M., & Hedge, S. U. (1989). Designing neural networks using genetic algorithms. *Proceedings of the Third International Conference on Genetic Algorithms and Their Applications,* Massachusetts Institute of Technology, 379–384.

Mountcastle, V. B. (1975). An organizing principle for cerebral function: The unit module and the distributed system. In G. M. Edelman & V. B. Mountcastle (Eds.), *The mindful brain.* Cambridge, MA: MIT Press.

Murre, J. M. J. (1992a). *Learning and categorization in modular neural networks.* Hemel-Hempstead: Harvester Wheatsheaf (Hillsdale, NJ: Lawrence Erlbaum).

Murre, J. M. J. (1992b). The effects of pattern presentation on interference in backpropagation networks. *Proceedings of the* 14*th Annual Conference of the Cognitive Science Society.* Hillsdale, NJ: Lawrence Erlbaum, 54–59.

Murre, J. M. J. (1993). Transputers and neural networks: An analysis of implementation constraints and performance. *IEEE Transactions on Neural Networks,* **4,** 284–292.

Murre, J. M. J., Phaf, R. H., & Wolters, G. (1989). CALM networks: A modular approach to supervised and unsupervised learning. *Proceedings of the International Joint Conference on Neural Networks Washington DC,* 1. New York: IEEE Press, 649–656.

Murre, J. M. J., Phaf, R. H., & Wolters, G. (1992). CALM: Categorizing and learning module. *Neural Networks,* **5,** 55–82.

Nelson, M. E., & Bower, J. M. (1990). Brain maps and parallel computers. *Trends in Neurosciences,* **13,** 403–408.

Pantle, A., & Sekuler, R. (1968). Size-detecting mechanisms in human vision, *Science,* **162,** 1146–1148.

Phaf, R. H. (1991). *Learning in natural and connectionist systems: Experiments and a model.* Unpublished doctoral dissertation, Leiden University, Leiden.

Phaf, R. H., Postma, E. O., & Wolters, G. (1990). ELAN-1: A connectionist model for implicit and explicit memory tasks. *Leiden Psychological Reports,* EP 01-90, Leiden.

Phaf, R. H., Van der Heijden, A. H. C., & Hudson, P. T. W. (1990). SLAM: A connectionist model for attention in visual selection tasks. *Cognitive Psychology,* **22,** 273–341.

Poggio, T., Gamble, E. B., & Little, J. J. (1988). Parallel integration of vision modules. *Science,* **242,** 436–440.

Posner, M. I. (1986). *Chronometric explorations of mind.* Oxford: Oxford University Press.

Posner, M. I., Peterson, S. E., Fox, P. T., & Raichle, M. E. (1988). Localization of cognitive operations in the human brain. *Science*, **240**, 1627–1631.

Prigogine, I., & Stengers, I. (1984). *Order out of chaos: Man's new dialogue with nature*. New York: Bantum.

Rakic, P., Bourgeois, J. P., Eckenhoff, M. F., Zecevic, N., & Goldman-Rakic, P. S. (1986). Concurrent overproduction of synapses in diverse regions of the primate cerebral cortex. *Science*, **232**, 232–235.

Ratcliff, R. (1990). Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological Review*, **97**, 285–308.

Rueckl, J. G., Cave, K. R., & Kosslyn, S. M. (1989). Why are 'what' and 'where' processed by separate cortical visual systems? A computational investigation. *Journal of Cognitive Neuroscience*, **1**, 171–186.

Rumelhart, D. E. (1977). Understanding and summarizing brief stories. In D. LaBerge & S. J. Samuels (Eds.), *Basic processes in reading: Perception and comprehension* (pp. 265–303). Hillsdale, NJ: Lawrence Erlbaum.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing. Volume 1: Foundations*. Cambridge, MA: MIT Press.

Rumelhart, D. E., & Zipser, D. (1985). Feature discovery by competitive learning. *Cognitive Science*, **9**, 75–112.

Shepherd, G. M. (1990). The significance of real neuron architectures for neural network simulations. In E. L. Schwartz (Ed.), *Computational neuroscience* (pp. 82–96). Cambridge, MA: MIT Press.

Shepherd, G. M., & Koch, C. (1990). Introduction to synaptic circuits. In G. M. Shepherd (Ed.), *The synaptic organization of the brain* (pp. 3–31). Oxford: Oxford University Press.

Solla, S. A. (1989). Learning and generalization in layered neural networks: The contiguity problem. In L. Personnas & G. Dreyfus (Eds.), *Neural networks: From models to applications* (pp. 168–177). Paris: I.D.S.E.T.

Szentágothai, J. (1975). The 'module-concept' in the cerebral cortex architecture. *Brain Research*, **95**, 475–496.

Szentágothai, J. (1977). The neuron network of the cerebral cortex: A functional interpretation. *Proceedings of the Royal Society of London, B*, **201**, 219–248.

Terzopoulos, D. (1986). Integrating visual information from multiple sources. In A. P. Pentland (Ed.), *From pixels to predicates: Recent advances in computational vision* (pp. 111–142). Norwood, NJ: Ablex.

Todd, J. T. (1985). The perception of structure from motion: Is projective correspondence of moving elements a necessary condition? *Journal of Experimental Psychology: Human Perception and Performance*. **11**, 689–710.

Vailliant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, **27**, 1134–1142.

Valdès, R. (1991). What is BioComputing? *DrDobb's Journal*, **16**(4), 46, 108–109.

Van Essen, D. C., Anderson, C. H., & Felleman, D. J. (1992). Information processing in the primate visual system: An integrated systems perspective. *Science*, **255**, 419–423.

von der Malsburg, C. (1973). Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, **14**, 85–100.

Walley, R. E., & Weiden, T. D. (1973). Lateral inhibition and cognitive masking: A neuropsychological theory of attention. *Psychological Review*, **80**, 284–302.

Warrington, E. K. (1982). The fractionation of arithmetical skills: A single case study. *Quarterly Journal of Experimental Psychology: Human Experimental Psychology*, **34**, 31–51.

Weber, B. H., Depew, D. J., Dyke, C., Salthe, S. N., Schneider, E. D., Ulanowicz, R. E., & Wicken, J. S. (1986). Evolution in thermodynamic perspective: An ecological approach. *Biology and Philosophy*, **4**, 373–405.

Whitley, D. (1989). The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. *Proceedings of the Third International Conference on Genetic Algorithms and their Applications*. Massachusetts Institute of Technology, 116–121.

Whitley, D., & Bogart, C. (1990). The evolution of connectivity: Pruning neural networks using genetic algorithms. *Proceedings of the International Joint Conference on Neural Networks, Washington DC*, 1. Hillsdale, NJ: Lawrence Erlbaum, 134–137.

Whitley, D., & Hanson, T. (1989). Towards the genetic synthesis of neural networks. *Proceedings of the Third International Conference on Genetic Algorithms and their Applications*. Massachusetts Institute of Technology, 391–396.

Whitley, D., & Starkweather, T. (1990). Optimizing small neural networks using a distributed genetic algorithm. *Proceedings of the International Joint Conference on Neural Networks, Washington DC*, 1. Hillsdale, NJ: Lawrence Erlbaum, 206–209.

Wickelgren, W. A. (1981). Human learning and memory. *Annual Review of Psychology*, **32**, 21–52.

Wolpert, D. H. (1990). The relationship between Occam's razor and convergent guessing. *Complex Systems*, **4**, 319–368.

Zadeh, L. A. (1987). *Fuzzy sets and applications: Selected papers*. New York: Wiley.

Zeki, S., & Shipp, S. (1988). The functional logic of cortical connections. *Nature*, **335**, 311–317.

APPENDIX

Method

The input of the networks consisted of black and white patterns, written on a computer screen by means of a computer mouse. The handwritten digits were converted into an input grid of 5×5 pixels. The proportional blackness of each square of the grid constituted the activation value. Each handwritten digit is, thus, represented by 25 continuous activation values. Three hundred stimuli were obtained by instructing 30 subjects to write the digits 0, 1, 2, . . . , 9. A random subset of 200 digits was selected and divided into a training set of 10 digits and a set of 190 digits used to assess generalization performance. During a genetic search, different sets were randomly compiled from the pool of 200 digits at regular intervals. The remaining 100 digits were not used to determine fitness values in any of the experiments but instead served to obtain an additional indication of the generalization performance of the best architecture solutions found by the genetic algorithm. Each of the 10 orthogonal patterns used for cueing stimulus categories consisted of 10 binary activation values (1000000000, 0100000000, . . . , 0000000001).

Each of the experiments reported relies on the application of a similar genetic algorithm. An initial population of 150 binary strings is randomly generated. Each string contains 45 1-bit genes that encode the modular configuration. A network contains a variable number of intermediate or hidden modules. In addition, it contains an input module of size 35 (25 nodes for handwritten digit input and 10 nodes for cueing categories) and an output CALM of size 10, both of which are a fixed part of a network and need not be coded in a string. Every module can be connected to every other module. As was explained near the beginning of this article, two modules A and B are connected by connecting all R-nodes in A to all R-nodes in B. These intermodular connections are always modifiable in CALM networks. The number of hidden modules, their sizes, and the intermodular connection structure are all encoded in the bitstring (and, thus, subject to genetic search). The maximum total number of R-nodes in all modules of the hidden CALMs is restricted to 20 that can be divided over a maximum of four hidden modules.

The assignment of the number of R-nodes to each hidden module is coded by the first 20 bits or genes of a string. The specific pattern of 0s and 1s defines the number of hidden modules and their sizes. Going through the string, starting at the first bit, a 0 signals the allocation of R-nodes to a new module with every next bit encountered. The last bit of a continuous substring of 1s signals the termination of allocating R-nodes to a module. Nonmeaningful combinations of

0s and 1s are ignored, which implies that less than the available 20 nodes might be allocated to hidden modules. For example, the string 10011000111101000000 allocates 13 R-nodes to three modules of size 4 (0011), 7 (0001111), and 2 (01). The applied method of coding module sizes can be likened to the use of area markers introduced by Harp et al. (1989). They used a fixed framework of markers with a variable number of genes within each marker area. We take a slightly different approach, using a fixed number of bits or genes and variable marker positions.

The remaining 25 bits of each string in the population are used to encode module connections. From the above allocation procedure it follows that the minimum size of a module is 2. A maximum of four hidden modules can be defined. Together with the input and output module this amounts to a maximum total of six modules that can be part of a network. Every module can receive connections from every other module with the exception of the input module, which can have only outgoing connections to other modules. Therefore, five fields of 5 bits are coding the incoming connections for all modules in a network. The first 5 bits are used for the output CALM, the remaining four fields code connections for the hidden modules. If less than four hidden modules have been defined, the corresponding last fields of 5 bits remain unused. Module connections are encoded in a straightforward way. The first bit of a field indicates whether a CALM is connected or not with the input module. The second bit indicates a connection from the output CALM, and the remaining 3 bits indicate connections from hidden modules. The last bits of every field are ignored when less than four hidden modules are defined. A connection from a module to itself is not allowed; bits encoding self-connections are ignored. It should be noted that the genetic algorithm has two different ways of not using network resources. Either not all 20 R-nodes are used to define hidden modules, or defined modules can remain unused by functionally disconnecting them from the network.

For the simulations of experiment 1 and 2 a change has been made to the CALM learning rule, concerning the dependency of the learning rate of the activation of the E-node. The E-node activation is a measure of the level of competition going on in a module. In a standard CALM the learning rate is (like the level of distributed noise) a linear function of the E-node activation. This function has been changed into a Gaussian function, such that high and low E-node activation values result in a low learning rate and at medium activation values, when competition is beginning to solve (i.e., there is something to learn), the learning rate is high.

The internal dynamics such as determined by the learning and activation rule parameters of the CALMs in a network can be an important determinant of overall performance. Therefore, it was decided to let the genetic algorithm search for a modular network configuration, and simultaneously optimize 17 floating point values that globally define the following parameters: the fixed weights of the eight types of internal connections of CALM, and nine learning and activation rule settings (see Murre, 1992a, for a detailed description of these parameters).

The parameter values are coded by a second string or chromosome containing 256 bits using seventeen 16-bit genes. Each gene encodes a parameter as a 16-bit number that is scaled to one of 65,535 (16 bits) possible floating point values within a range specific to that parameter. In contrast to the strings coding the network architecture, which are generated completely at random, mutated standard values were used for the parameter strings as initial values. The 14 standard values were inversely scaled to 16-bit binary values that were mutated by randomly flipping on average 1 in every 5 bits to produce a population of 150 strings. The standard parameters of CALM provide a set of values that has proved to meet the demands of a wide variety of tasks and it may, therefore, be expected to be not too far from a good solution to any specific learning task. Hence, using completely random values would unnecessarily slow down genetic search.

A coded solution provided by the genetic algorithm thus consists of two strings, one coding a modular configuration and another coding global parameter values. Both strings are subject to the same selection and replacement operator and are in this sense linked. Crossover and mutation are, however, applied to each string individually, whereby the parameter chromosome of one solution is never crossed with the architecture chromosome of another solution and vice versa. The two strings or chromosomes can be viewed as separate higher order building blocks of a single genetic solution.

The *static population model* (Whitley, 1989) was used for selection and replacement of solutions. Apart from rank-based selection this model uses *one-at-a-time selection and replacement*. The algorithm applied crossover to both the architecture and parameter string with a crossover probability of 0.2 and 1.0, respectively. This implies that two selected parameter strings are always crossed while their linked architecture strings are passed on uncrossed to the offspring in 80% of the reproduction trials. Inversion was applied to both strings with equal probability (1.0). Mutation probabilities again differed for the parameter string (0.001) and the architecture string (0.02). Because the different parts of a network structure are functionally much interrelated, crossing different network structures was not expected to be very effective (this problem can be overcome by using different, grammar-based coding schemes for architectures). The formation of new architectures was, therefore, largely left to the introduction of small changes in existing architecture solutions through mutation. An important deviation from the standard static population model was introduced by a continuous reevaluation of solutions already in the population. A large part of the variance in the obtained fitness values resulted from the use of many different training set/test set combinations. Therefore, each solution was regularly retested on different sets. Furthermore, only once in every 150 network evaluations a new training set and test set were randomly selected from the pool of 200 digits, with each block of 150 networks contained 100 new solutions and 50 retests of solutions. As a result, the performance of a large number of network architectures is continuously compared on the same training and test sets. This specific evaluation scheme considerably improved the genetic search for architecture solutions that are optimal for many different training and test set combinations. In addition, apart from fitness values on specific sets, the quality of a solution could be assessed by the time it stayed in the population through monitoring the number of retests it had undergone.