Neural Systems and Artificial Life Group,
Institute of Psychology,
National Research Council, Rome

Evolving Modular Architectures for Neural Networks

Andrea Di Ferdinando, Raffaele Calabretta and Domenico Parisi

September 14, 2000
(revised October 24, 2000)

In: R. French & J. Sougné (Eds.), *Proceedings of the sixth Neural Computation and Psychology Workshop: Evolution, Learning, and Development*, pp. 253-262, London: Springer Verlag.

Department of Neural Systems and Artificial Life
Institute of Psychology, Italian National Research Council
V.le Marx, 15 00137 Rome - Italy
Phone: +39-06-860901, Fax: +39-06-824737
E-mail: {andread, rcalabretta, parisi}@ip.rm.cnr.it
http://gral.ip.rm.cnr.it

Evolving Modular Architectures for Neural Networks

Andrea Di Ferdinando, Raffaele Calabretta & Domenico Parisi
Institute of Psychology
National Research Council, Rome, Italy
{andread, rcalabretta, parisi}@ip.rm.cnr.it

Abstract

Neural networks that learn the What and Where task perform better if they possess a modular architecture for separately processing the identity and spatial location of objects. In previous simulations the modular architecture either was hardwired or it developed during an individual's life based on a preference for short connections given a set of hardwired unit locations. We present two sets of simulations in which the network architecture is genetically inherited and it evolves in a population of neura1 networks in two different conditions: (1) both the architecture and the connection weights evolve; (2) the network architecture is inherited and it evolves but the connection weights are learned during life. The best results are obtained in condition (2). Condition (1) gives unsatisfactory results because (a) adapted sets of weights can suddenly become maladaptive if the architecture changes, (b) evolution fails to properly assign computational resources (hidden units) to the two tasks, (c) genetic linkage between sets of weights for different modules can result in a favourable mutation in one set of weights being accompanied by an unfavourable mutation in another set of weights.

1. Modularity as a Solution to the Problem of Neural Interference

Neural networks that learn only one task can have simple architectures and may not need modularity. However, real organisms generally have not one task but many different tasks to accomplish in order to survive and reproduce. Hence, their nervous systems tend to be organized with anatomically and functionally distinct modules. Using neural networks that have to learn different tasks can help understand why organisms develop modular nervous systems.

Why are neural modules useful? One possible answer is that modules allow a neural network to solve the problem of neural interference. Learning consists in progressively modifying an initial set of weights in such a way that at the end of learning the network produces the desired output in response to each input. Consider a single one of these weights. During learning the weight’s initial value is gradually changed in such a way that at the end of learning it will be the correct value, i.e., the weight value that together with the values of the other connection

weights produces the correct output. If the network has only one task to learn, the problem can be solved reasonably easily. However, if the network has to learn two different tasks and the particular connection weight we are considering has a role in both tasks, i.e., the output the network must generate in response to the input depends on the value of this weight both when the network is engaged in one task and when it is engaged in the other task, then the situation may become more complicated. The correct accomplishment of the first task may require that the initial weight value of the connection be increased during learning while the correct accomplishment of the second task may require that the initial value be decreased. This will lead to some sort of interference or conflict between the two tasks. (The problem encountered by nonmodular architectures learning multiple tasks is called "cross-talk" by Plaut and Hinton [11] and Jacobs *et al*. [7]. Cross-talk refers to contradictory messages arriving to a neural network's hidden unit, interference to contradictory messages arriving to a network's connection weights. But the two are more or less equivalent.)

Modularity solves the interference problem. If the network architecture is such that no single connection weight has a role in determining the network's output for both tasks, there will be no interference between the two tasks. The weight value of each particular connection will be changed during learning to satisfy the requirements of the single particular task in which the connection plays a role and it will never happen that the same connection will have to respond to contradictory pressures to change its weight value. All the connections that play a role in one particular task and in no other task constitute a module. The connections of a module are "proprietary", i.e., they are dedicated exclusively to the accomplishment of a single task. Their weight value can be adjusted during learning without interfering with, and being interfered by, other tasks.

An example of the problem of multiple tasks is represented by organisms that must recognize both the identity (What) and the spatial location (Where) of visually perceived objects. Nervous systems that must learn this What and Where task have two separate neural pathways, a ventral (temporal) pathway for recognizing the identity of the object and a dorsal (parietal) pathway for identifying its location [14]. (The dorsal pathway can be concerned with "How" to accomplish a physical movement with respect to the object rather than with "Where" the object is, but the two interpretations can be considered as equivalent for our purposes.) Rueckl *et al*. [13] have taught the What and Where task to both modular and nonmodular networks using the backpropagation procedure and have found that modular networks learn much better the task than nonmodular ones. In Rueckl *et al*.'s simulations the network architectures are hardwired by the researchers and what is investigated is how different network architectures give different results. In biological reality it is not the researcher but nature that creates network architectures. Hence, it might be interesting to study how modular network architectures may spontaneously arise as part of a process of development in individual networks or evolution in a population of networks.

Jacobs and Jordan [8] have described simulations in which modular architectures for the What and Where task emerge as part of an individual's development. Their

model is based on a preference for establishing short rather than long connections between pairs of neurons during brain development. During development individual neurons reach particular positions in the physical space of the brain. When the neurons grow their axons and establish connections with other neurons, it is more probable that a connection will be established between two spatially close neurons than between two more distant neurons. Using this simple developmental rule they were able to show that modular architectures rather than nonmodular ones tend to emerge as a result of the development of the brain. However, Jacobs and Jordan [8] seem to be able to obtain this result only because they hardwired the spatial location of units in such a way that separate modules for the What and the Where task tend to emerge developmentally. In other words, in their model nature has replaced the researcher only partially. Modular architectures emerge because of decisions taken by the researcher, not truly spontaneously. One could simulate the entire process of the emergence of modular architecture during brain development by using a genetic algorithm to find out evolutionarily the appropriate locations of units in the physical space of the nervous system and then have the preference for short connections generate the appropriate network architecture during development. This would be more appropriately called development since it would consist in changes during life in which inherited genetic information has a critical role. (For a simulation of brain development in which both the physical location of individual neurons and the establishment of connections, especially short connections, between neurons emerge spontaneously, see [4]).

Another possibility is to imagine that biological evolution takes care of the problem of finding the appropriate modular architecture. Networks that must learn two distinct tasks are born with a genetically inherited modular architecture which has been shaped during the course of evolution. Network architecture emerges not during an individual's life but during a succession of generations in a population of individuals. (Murre [10] also has suggested to use the genetic algorithm to design modular network architectures.)

We have conducted two sets of simulations using the genetic algorithm as a model of evolution to develop neural networks that are able to accomplish the What and Where task. In a first set of simulations we used the genetic algorithm to evolve both the network architecture and the connection weights but we were unable to solve the task using this approach. In a second set of simulations the genetic algorithm was used to evolve the network architecture but the connection weights were learned by the individual networks during their 'life' using the backpropagation procedure. This second approach gave the desired solution.

2. The What and Where Task

The What and Where task requires a neural network to recognize both the identity and the spatial location of perceived objects. In Rueckl *et al.* [12] the neural network is presented in each cycle with one of 9 different objects that can appear in one of 9 different positions on a retina for a total of 9x9=81 possible inputs. The network has two distinct sets of 9 output units each for indicating the identity and the location of

the presented object, respectively, and a single layer of 18 hidden units. In the nonmodular architecture all the hidden units are connected with both the What output units and the Where output units. Various modular architectures are tried out. The modular architecture that performs much better than the nonmodular architecture has 14 hidden units connected only with the What output units and the remaining 4 hidden units connected only with the Where output units (Figure 1). The reason for the success of this particular architecture is that the What subtask is more difficult than the Where subtask. The networks learn using the backpropagation procedure.

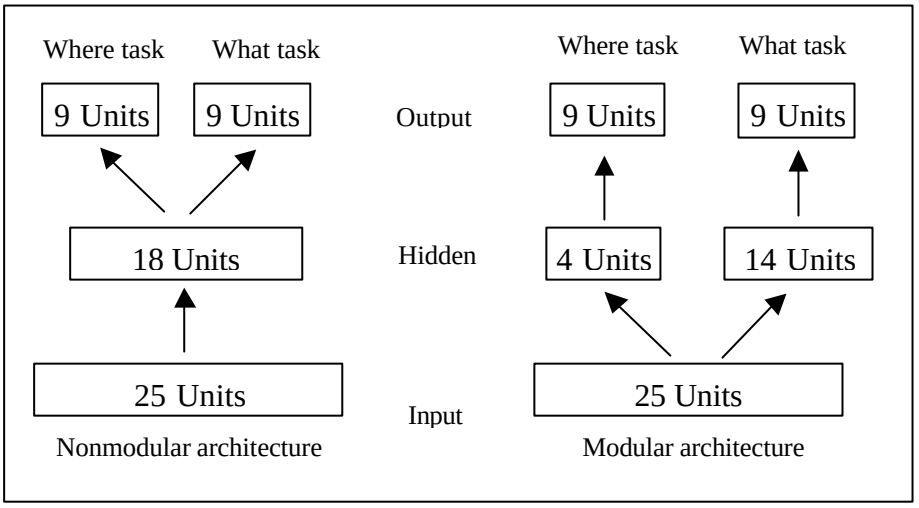

Figure 1: Nonmodular (left) and modular (right) network architectures for the What and Where task.

In Rueckl *et al.*'s simulations the network architecture is imposed by the researcher. The networks' task is to find the appropriate connection weights given a certain architecture. However, the What and Where task could also be solved at the population level. Imagine an entire population of networks, each different from all the others. Individual networks are born with genetically inherited information that specifies the network architecture and, possibly, also the connection weights. This genetically inherited infomation is the result of a process of biological evolution which takes place in successive generations of individuals and is based on the selective reproduction of the most successful individuals and the constant addition of new variants to the population's genetic pool.

We describe two sets of simulations. In the first set both the network architectures and the connection weights evolve and are genetically inherited. In the second set the network architectures evolve and are genetically inherited but the appropriate connection weights are learned during life by each individual.

3. Simulations

3.1 Using the Genetic Algorithm to Evolve Both the Network Architecture and the Connection Weights

Imagine a population of organisms living in an environment in which the reproductive chances of each individual depend on the individual's performance in the What and Where task. The individuals that have a smaller error on the What and Where task are more likely to reproduce than the individuals with a larger error.

An individual is born with an inherited genotype which is divided up into two parts. One part specifies the architecture of the individual's neural network and the other part the network's connection weights. Some general features of the architecture are fixed and identical in all individuals (and therefore are not encoded in the genotype and do not evolve). All architectures have three layers of units with 25 input units (encoding a 5x5 retina), 18 hidden units, and 18 output units (9 for indicating the identity of the perceived object and 9 for indicating the object's spatial location in the retina). In all architectures each input unit is connected with all the hidden units. What can vary from an architecture to another are the connections between the hidden units and the output units. The portion of the genotype which encodes the network architecture contains 18 genes, one for each hidden unit. Each of these architectural genes has three possible values that specify if the corresponding hidden unit is connected (a) to all the What output units, (b) to all the Where output units, or (c) to both the What and the Where output units. The third possibility, (c), is included to allow for the evolution of nonmodular architectures. The other part of the genotype encodes the connection weights and it includes one gene for each possible connection weight (weight genes). The weight genes are 774 because there is a maximum of 774 connection weights in a nonmodular network. Modular architectures have less than 774 genes and in this case some of the weights may remain unexpressed. The weight genes are encoded as real numbers.

At the beginning of the simulation a population of 100 individuals is created and each individual possesses a genotype with random values for both the architectural and the weight genes. The values of the weight genes are randomly chosen in the interval between -0.3 and +0.3. Each individual is presented with the 81 input patterns of the What and Where task and an individual's fitness is greater the lower its summed squared error on these patterns. The 20 best individuals are selected for reproduction. Each of these individuals generates 5 offspring which inherit the genotype of their single parent with the addition of some random mutations. The architectural genes are mutated by replacing the value of a gene with a new randomly chosen value with a probability of 5%. The weight genes are mutated by adding a quantity randomly chosen in the interval between -1 and +1 to 10% of the genes. The simulation is terminated after 10,000 generations. Ten replications of the simulation were run with randomly chosen initial conditions.

The results of the experiment show that the genetic algorithm is unable to solve the What and Where task if both the architecture and the connection weights are subject to evolution and are genetically inherited. The total error is about 40 after 10,000 generations. In Rueckl *et al.*'s simulations using the backpropagation

procedure the terminal error is practically zero for the best architecture. While the performance in the Where task is good enough but not as good as in Rueckl *et al.* (error = 7), the performance in the What task is very poor (error = 33).

These negative results appear to be caused by the difficulty on the part of the genetic algorithm to evolve an appropriate set of weights if the network architecture is evolving at the same time. Each architecture has its own appropriate set of weights and, therefore, changing an architecture can be destructive from the point of view of the weights. A given set of weights which is appropriate for a given architecture may be completely inappropriate if the architecture changes. In our simulations the genotype specifies the weights of all possible connections even if some of these connections are not expressed in the phenotype. Therefore, when an unexpressed connection get expressed as a result of a mutation, its value is not zero. But adding even a single connection with its value already specified can destroy the equilibrium of the connectivity pattern. The same applies if the weight value of the new connection is randomly generated or if a previously expressed connection is canceled by a mutation together with its connection weight.

This interpretation is supported by the results obtained by manipulating the mutation rate. If the mutation rate of the network architecture is increased (10%), the final error increases (60). If it is reduced (1% and even 0.1%) the final error decreases although it never approaches zero (25 for 1% mutation rate; 22 for 0.1% mutation rate).

However, the bad results of these simulations may be due to another reason in addition to the disruption caused by the addition or deletion of connections with their weight value. One would expect that the network architecture that eventually evolves is the architecture that Rueckl *et al.* [13] have found is the best architecture, that is, an architecture with more hidden units dedicated to the more complex What task than to the simpler Where task. This is not the case in our simulations. Although the genetic algorithm does evolve modular architectures (on the average only about 2 hidden units are connected to both the What output units and the Where output units), the network architecture which tends to evolve has more hidden units assigned to the Where task than to the What task. It is not surprising then that the networks' performance on the total task is not good.

The reason for the failure of the genetic algorithm to evolve the appropriate modular architecture seems to be that in the initial generations the algorithm concentrates on the easier task, the Where task, and dedicates many computational resources (hidden units) to this task. When the performance on this task is almost perfect, however, the algorithm is unable to shift computational resources from the Where task to the more difficult What task. More specifically, in the earlier generations the individuals that are selected for reproduction are those that are good at the Where task even if they are not very good at the What task. These individuals tend to have network architectures with more hidden units assigned to the Where task (which decides if they reproduce or not) than to the What task. When in the later generations competition becomes harsher and selection would reward individuals that are good both at the Where task and at the What task, the random genetic mutations are unable to modify a situation in which most hidden units are

already assigned to the Where task and evolution is unable to produce individuals that are good at both tasks.

	Hidden units			Error		
Runs	Where	What	Both	Where	What	Total
1	15.8	0.7	1.5	5.5	39.1	44.7
2	6.0	5.1	6.9	4.8	28.7	33.4
3	**3.2**	**11.3**	**3.5**	**9.9**	**27.3**	**37.1**
4	**7.2**	**9.4**	**1.4**	**8.9**	**27.6**	**36.6**
5	16.4	0.3	1.3	5.8	42.2	48.0
6	8.0	5.4	4.5	6.4	31.1	37.4
7	**6.9**	**10.8**	**0.3**	**7.5**	**25.6**	**33.0**
8	8.0	6.5	3.5	8.4	28.6	37.0
9	14.8	2.1	1.1	5.9	41.0	46.9
10	14.8	2.4	0.7	8.2	44.3	52.6

Table 1: Number of hidden units allocated to the Where task, to the What task, and to both tasks, and error on the Where task, the What task, and total error for each of 10 replications of the simulation (average of 100 individuals for each replication). In 3 replications (bold face) more hidden units are allocated to the What task than to the Where task and still the performance is not good.

However, even this may not be the entire story. If we look at Table 1, we see that at least in some replications of the simulation (3 out of 10) more hidden units are correctly dedicated to the What task rather than to the Where task. But even in these replications of the simulation the error on the What task, and therefore also the total error, remains quite high. Hence, the failure of the genetic algorithm to produce efficient networks for the What and Where task appears to be due to its inability to select the appropriate connection weights even for networks which have the appropriate modular architecture.

This may reveal a general inability of genetic algorithms of the type we used in our simulations to evolve the appropriate connection weights for modular networks. We have run an additional set of simulations (not reported here) in which the genetic algorithm tries to find the appropriate connection weights given a fixed modular architecture of the appropriate type, with little success. The reason seems to be that, since the connection weights of different modules are separately encoded in the genotype, a favourable mutation of the connection weights of one module can be accompanied by a nonfavourable mutation of the connection weights of another module, with little total advantage. This seems to be a form of genetic linkage. Either the individual in which the two mutations occur is selected for reproduction - and in this case the nonfavourable mutation in the second module becomes part of the population's pool -, or the individual is not selected for reproduction - and in this case the favorable mutation in the first module is lost.

3.2 Using the Genetic Algorithm to Evolve the Network Architecture and the Backpropagation Procedure for Learning the Connection Weights

Perhaps, then, the solution to the various problems we have seen in the simulations described so far is to use the genetic algorithm to evolve the appropriate network architecture at the population level and the backpropagation procedure to have each individual network learn the connection weights for its inherited network architecture during life. We have run a second set of simulations in which the inherited genotypes encode a variety of possible network architectures but the genotype does not encode the connection weights. The connection weights are not genetically inherited but they are learned during life. At birth each individual is assigned a random set of weights for the particular network architecture it inherits and then the individual learns to do the What and Where task exactly as in the Rueckl *et al.*'s simulations. At the end of learning the terminal error of each individual determines the individual's reproductive chances.

Hidden units			Error		
Where	What	Both	Where	What	Total
4.7	12.2	1.1	0.0	1.6	1.6

Table 2: Number of hidden units allocated to the Where task, to the What task, and to both tasks, and error on the Where task, the What task, and total error for the average individual in the last generation (average of 10 replications of the simulation).

The results are that, first, at the end of the simulation the terminal error is near zero for both the What and the Where tasks - even if it is still somewhat larger for the What task - and, second, the evolved network architectures tend to be the optimal architectures with more hidden units dedicated to the What task than to the Where task (Table 2).

4. Discussion

A neural network that must acquire a capacity to do more than one tasks is better able to acquire this capacity if the network architecture is modular because neural modules prevent the occurrence of neural interference, defined as the arrival of contradictory messages during learning for changing the value of connection weights involved in more than one task. Neural modules contain connection weights involved in only one task and therefore they avoid neural interference.

Neural modules can be hardwired by the researcher or they can spontaneously emerge during development or evolution. Using the genetic algorithm as a model of biological evolution we have simulated the evolution of network architectures that are appropriate for recognizing both the identity and the spatial location of perceived

objects. Modular and nonmodular architectures compete in the successive generations of a population of neural networks and modular architectures should emerge as the winning ones.

We have compared two conditions, one in which both the network architecture and the connection weights evolve and are genetically inherited and another one in which only the architecture evolves and is inherited while the connection weights are learned during an individual's life. Only the second condition produces satisfying results, that is, the appropriate modular architecture and high levels of performance in the task. If both the architecture and the connection weights are encoded in the genotype, a change in the network architecture with the addition or deletion of even a single connection can suddenly make a set of weights that has evolved with the preceding architecture inappropriate. Furthermore, evolution may not be the best method for evolving the connection weights for modular networks because a favorable genetic mutation in one module may be accompanied by an unfavorable mutation in another module, although sexual recombination or genetic duplication [3] might help solve this problem.

As suggested by various authors (see, for example, [2]), cooperation between evolution and learning can be the best solution to the problem of acquiring complex capacities, compared with having either evolution or learning completely take care of the problem. However, it is not only that evolution and learning must cooperate and both have a role in the acquisition process but the best solution might be to have evolution take care of the network architecture and learning of the connection weights. Hence, the network architecture is genetically inherited but the connection weights are not. They are learned during life. (This solution has been proposed on the basis of more general considerations by Elman *et al.* [5]).

One should not, however, overemphasize this particular type of division of labor between evolution and learning. A number of other arrangements may exist that maintain the general scheme of entrusting the network architecture to evolution and the connection weights to learning but distribute the details of this scheme differently. For example, the initial connection weights can be encoded in the genotype and then learning modifies these initial weights. It has been shown that the initial weights influence learning [9] and that evolution may find out what are the best initial weights for learning some particular task (see, for example, [1]). Other schemes may involve the genetic encoding not of the actual connection weights themselves but only of various constraints on the connection weights. For example, whether some particular connection weight is positive (excitatory) or negative (inhibitory) may be encoded in the genotype but it is learning that finds out what is the most appropriate absolute value for the weight. Or the range of variation of the value of some weight may be encoded in the genotype, but the actual value within this range is identified by learning. Or, again, evolution can find the appropriate learning parameters and learning the actual weight values [6]. On the other side, learning can change an inherited network architecture by adding and/or deleting connections (cf. the pruning and tiling algorithms [12]). However, it could still be the general case that evolution identifies the general layout of a species' brain and learning refines what is inherited by adjusting the weights on the brain's connections.

References

1. Belew, R. K., McInerney, J., & Schraudolph, N. (1991). Evolving networks: using the genetic algorithm with connectionist learning. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (eds), *Artificial Life II*. Addison-Wesley, Reading, MA.
2. Belew, R. K. & Mitchell, M. (1996). *Adaptive Individuals in Evolving Populations.* Addison-Wesley, Reading, MA.
3. Calabretta, R., Nolfi, S., Parisi, D. & Wagner, G. P. (2000). Duplication of modules facilitates the evolution of functional specialization. *Artificial Life* 6:69-84.
4. Cangelosi A., Parisi D. & Nolfi S. (1994). Cell division and migration in a 'genotype' for neural networks. *Network* 5:497-515.
5. Elman, J. L., Bates, E. A., Johnson, M. H., Karmiloff-Smith, A., Parisi, D. & Plunkett, K. (1996). *Rethinking innateness. A connectionist perspective on development*. The MIT Press, Cambridge, MA.
6. Floreano, D. & Urzelai, J. (2000). Evolutionary robots with on-line self-organization and behavioral fitness. *Neural Networks* 13:431-443.
7. Jacobs, R. A., Jordan, M. I. & Barto, A. G. (1991). Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive Science* 15:219-250.
8. Jacobs, R. A. & Jordan, M. I. (1992). Computational consequences of a bias toward short connections. *Journal of Cognitive Neuroscience* 4:323-335.
9. Kolen J. F. & Pollack, J. B. (1990). Back-propagation is sensitive to initial conditions. *Complex Systems* 4:269-280.
10. Murre, J. M. J. (1992). *Learning and categorization in modular neural networks.* Harvester, New York, NY.
11. Plaut D. C. & Hinton, G. E. (1987). Learning sets of filters using back-propagation. *Computer Speech and Language* 2:35-61.
12. Reed, R. D. & Marks II, R. J. (1999). *Neural Smithing. Supervised Learning in Feedforward Artificial Neural Networks.* The MIT Press, Cambridge, MA.
13. 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.
14. Ungerleider, L G. & Mishkin, M. (1982). Two cortical visual systems. In D. J. Ingle, M. A. Goodale & R. J. W. Mansfield (Eds.), *The Analysis of Visual Behavior.* The MIT Press, Cambridge, MA.