Task Decomposition Through Competition in a Modular Connectionist Architecture

Robert A. Jacobs

Department of Computer & Information Science

University of Massachusetts, Amherst, MA 01003

COINS Technical Report 90–44

May 1990

TASK DECOMPOSITION THROUGH COMPETITION IN A MODULAR CONNECTIONIST ARCHITECTURE

A Dissertation Presented by

ROBERT A. JACOBS

Submitted to the Graduate School of the
University of Massachusetts in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 1990

Department of Computer and Information Science

© Copyright by Robert A. Jacobs 1990

All Rights Reserved

TASK DECOMPOSITION THROUGH COMPETITION IN A MODULAR CONNECTIONIST ARCHITECTURE

A Dissertation Presented by

ROBERT A. JACOBS

Approved as to style and content by:

Andrew G. Barto, Chair of Committee

Roderic A. Grupen, Member

Christopher V. Hollot, Member

Michael I. Jordan, Member

W. Richards Adrion, Department Chair
Computer and Information Science

ACKNOWLEDGEMENTS

The influence of Andrew Barto and Michael Jordan on my thinking is evidenced throughout this dissertation. They provided technical training, constructive criticism, guidance, and inspiration throughout my tenure as a graduate student. Andy helped me to discriminate between vital research topics and those of secondary import, and urged me to focus my energies on those issues that should be central to learning theory in the future. Andy also taught me to organize my thoughts and communicate them in a coherent manner. Mike showed me the ingredients that go into scientific research at the day to day level. By allowing me to observe his own activities, Mike attempted to teach me how to properly define a research question and how to evolve and refine one's thinking so that steady progress is made towards answering that question. Mike stressed the importance of translating my abstract, heuristic notions into concrete, mathematical statements. Furthermore, Mike was always willing to study my research at a detailed level, to commend those aspects that were of interest, and to challenge those aspects that required further thinking.

In addition to Andy and Mike, I wish to thank the other members of my committee, Roderic Grupen and Christopher Hollot. Their questions forced me to re–evaluate and clarify several points in this thesis.

This research has benefited from conversations with many people who have been willing to share their thoughts and suggestions. I am particularly grateful to Jonathan Bachrach, Kyle Cave, Vijaykumar Gullapalli, Geoffrey Hinton, Stephen Kosslyn, Michael Mozer, Steven Nowlan, and Richard Sutton.

Much of the research reported here was conducted during a nine month stay at the Massachusetts Institute of Technology where I was a visiting graduate student in the Department of Brain & Cognitive Sciences. I thank the members of that department for welcoming me into their community and for allowing me to use their facilities.

Portions of this thesis appeared in a paper that I co–authored with Michael Jordan and Andrew Barto (Jacobs, Jordan, and Barto [32]). Consequently, some of the thoughts and writing contained in this thesis result from a collaboration between myself, Jordan, and Barto. Some of the ideas in Section 3.5 were contributed by Steven Nowlan and Geoffrey Hinton who convinced me of the soundness of their arguments by conducting a number of computer simulations using a modified version of the modular architecture that I propose here. Charles Anderson, Judy Franklin, and Stephen Judd contributed LaTeXmacros that I used to format this thesis.

This research was supported by funding provided to Andrew Barto by the Air Force Office of Scientific Research, Bolling AFB, under Grant AFOSR-89-0526, and

the National Science Foundation under Grant ECS-8912623. My stay at MIT was supported in part by funding provided to Michael Jordan from the Siemens Corporation.

ABSTRACT

TASK DECOMPOSITION THROUGH COMPETITION IN A MODULAR CONNECTIONIST ARCHITECTURE

SEPTEMBER 1990

ROBERT A. JACOBS, B.A., UNIVERSITY OF PENNSYLVANIA

M.S., UNIVERSITY OF MASSACHUSETTS

PH.D., UNIVERSITY OF MASSACHUSETTS

Directed by: Professor Andrew G. Barto

A novel modular connectionist architecture is presented in which the networks composing the architecture compete to learn the training patterns. As a result of the competition, different networks learn different training patterns and, thus, learn to compute different functions. The architecture performs task decomposition in the sense that it learns to partition a task into two or more functionally independent

tasks and allocates distinct networks to learn each task. In addition, the architecture tends to allocate to each task the network whose topology is most appropriate to that task, and tends to allocate the same network to similar tasks and distinct networks to dissimilar tasks. Furthermore, it can be easily modified so as to learn to perform a family of tasks by using one network to learn a shared strategy that is used in all contexts along with other networks that learn modifications to this strategy that are applied in a context sensitive manner. These properties are demonstrated by training the architecture to perform object recognition and spatial localization from simulated retinal images, and to control a simulated robot arm to move a variety of payloads, each of a different mass, along a specified trajectory. Finally, it is noted that function decomposition is an underconstrained problem and, thus, different modular architectures may decompose a function in different ways. A desirable decomposition can be achieved if the architecture is suitably restricted in the types of functions that it can compute. Finding appropriate restrictions is possible through the application of domain knowledge. A strength of the modular architecture is that its structure is well–suited for incorporating domain knowledge.

TABLE OF CONTENTS

Acknowledgements iv

Abstract vii

List of Tables xii

List of Figures xiii

Chapter

1 Introduction 1

1.1 Outline of the Thesis 6

2 Advantages of Modular Connectionist Architectures 8

2.1 Learning Speed 9

2.2 Generalization 13

2.3 Representation 14

2.4 Hardware Constraints 16

3 A Modular Connectionist Architecture 20

3.1 Output of the Architecture 20

3.2 Training of the Architecture 21

3.3 Discussion 25

3.4 Relationship to Previous Research 27

3.5 An Alternative Error Function 34

4 The What and Where Vision Tasks 42

4.1 Simulation Experiments 45

4.1.1 Temporal Crosstalk 46

4.1.2 Spatial Crosstalk 54

5 Multi–Payload Robotics Task 63

5.1 Training Procedure 64

5.2 Comparison of Architectures 70

5.3 Utilization of Domain Knowledge 86

6 Task Decomposition and Network Architectures 92

6.1 Simulations 93

6.2 Design of Modular Architectures 101

7 Conclusions 103

7.1 Future Work 104

Appendices

A Simulation Details for Chapter 4 110

B Simulation Details for Chapter 5 113

C Simulation Details for Chapter 6 116

References 118

LIST OF TABLES

4.1 Systems studied by Rueckl, Cave, and Kosslyn on the "what" and "where" tasks. 44

4.2 Systems used in the temporal crosstalk experiments. 48

4.3 The modular architecture used in the spatial crosstalk experiment. . . 59

5.1 Parameters of the robot arm. 69

5.2 f_i and g_i are the transformations of the robot arm's joint positions, velocities, and accelerations used in the input vectors of sets L and NL respectively. The subscripts on $\theta, \dot{\theta}$, and $\ddot{\theta}$ are the joint number. . 71

5.3 Single networks, modular architectures, and modular architectures with share networks used in the multi–payload robotics experiments. . . . 73

5.4 Three simulations that make use of a priori knowledge of the domain. 87

6.1 Systems used in the sine wave experiments. 95

A.1 Parameter values used in the temporal crosstalk experiments. 111

B.1 Parameter values used in the multi–payload robotics experiments. . . 115

LIST OF FIGURES

2.1 A: A network that is susceptible to spatial crosstalk. B: A modular architecture that is not susceptible to spatial crosstalk. 11

3.1 A modular connectionist architecture. 21

3.2 Multiplicative connection—The input to unit C is the product of the activations of units A and B. 31

3.3 Arrangement of networks used by the modular and cooperative architectures. 39

4.1 Learning curves for the 26 $\rightarrow$ 18 $\rightarrow$ 9 network on the "what" and "where" tasks using random and blocked training. 49

4.2 Learning curves for the 26 $\rightarrow$ 36 $\rightarrow$ 9 network on the "what" and "where" tasks using random and blocked training. 51

4.3 The modular architecture simulated in the temporal crosstalk experiments. 52

4.4 Learning curves for the modular architecture on the "what" and "where" tasks using random and blocked training. 55

4.5 A modular architecture with multiple gating networks ($\mathbf{y}_1^{[1\cdots m]}$ denotes the vector whose components are the first m components of expert network 1's output vector, and the other expressions similarly denote subvectors). 57

5.1 A: During the performance stage, both a feedforward controller and feedback controller compute torques used to control the robot arm. B: During the training stage, the feedforward controller receives the actual joint variables. Its desired output is the torque applied to the arm. 67

5.2 Two–joint planar arm. 69

5.3 A modular architecture with a share network. 75

5.4 Learning curves for the systems trained with the input vectors from set L. 78

5.5 Allocation of modular architecture MA–L's expert networks to payloads. 79

5.6 Allocation of modular architecture with a share network MAS–L's expert networks to payloads. 80

5.7 Learning curves for the systems trained with the input vectors from set NL. 83

5.8 Allocation of modular architecture MA–NL's expert networks to payloads. 84

5.9 Allocation of modular architecture with a share network MAS–NL's expert networks to payloads. 85

5.10 Learning curves for the three simulations during the last stage of training. 88

6.1 Graph of f. 94

6.2 Task decomposition discovered by System 1. 96

6.3 Task decomposition discovered by System 2. 99

6.4 Task decomposition discovered by System 3. 100

CHAPTER 1

INTRODUCTION

Although many biologists and psychologists share the view that the brain has a modular architecture, there is no general agreement on the number of modules, the function of the modules, the nature of the interaction between modules, or the manner in which the modules develop. One reason for this diversity of opinion is that answering questions about the modular nature of the brain involves the difficult task of reasoning about a system with a large number of interacting components. Even systems of interacting components with a small fraction of the brain's complexity present formidable conceptual and analytical difficulties. In many cases, mathematical and computer models provide essential tools for understanding aspects of these systems. One class of models that has the potential for helping to answer questions about modular systems is the class of connectionist models, also known as artificial neural network models.

A hierarchical classification of the components of connectionist models may be defined in which a unit is the finest level of classification, a layer is a coarser level,

and a network is a still coarser level. Connectionist researchers typically design systems that are modular at the level of units or layers. In this thesis we argue that there are significant practical and theoretical advantages to be realized by considering modularity at the level of networks. In other words, we argue that connectionist architectures should consist of multiple networks, and that connectionist learning algorithms should be designed to take advantage of this modular structure.

Although terms such as *layer* or *network* are imprecise, it is generally agreed that they provide a convenient language for discussing connectionist architectures. An analogous situation occurs in the neurosciences where researchers debate whether nervous systems are best conceptualized at the level of the neuron or at coarser levels such as the column, hypercolumn, or area (Kaas [36]). This debate persists despite the lack of agreement on the precise definitions of the coarser taxonomic levels. For our purposes, we rely on the intuition that portions of an architecture larger than a unit or layer that learn and perform different functions constitute different networks.

This thesis introduces a novel modular connectionist architecture in which the networks composing the architecture compete to learn the training patterns. An outcome of this competition is that different networks learn different training patterns and, thus, learn to compute different functions. The architecture performs task decomposition in the sense that it learns to partition a task into two or more functionally independent tasks and allocates distinct networks to learn each task. In addition, it tends to allocate to each task the network whose topology is most appropriate to that task, and tends to allocate the same network to similar tasks and distinct networks

to dissimilar tasks. Furthermore, it can be easily modified so as to learn to perform a family of tasks by using one network to learn a shared strategy that is used in all contexts along with other networks that learn modifications to this strategy that are applied in a context sensitive manner. These properties are highlighted in this introductory chapter and detailed in the remainder of the thesis.

Task decomposition can take at least two different forms. A complex task may be decomposed into a set of simpler tasks that must be performed in sequential order (Sutton [68], Barto, Sutton, and Watkins [11]). Alternatively, a complex task may be decomposed into a set of simpler tasks such that different tasks need to be performed in different contexts. Mathematically, this is equivalent to saying that a complex function may be decomposed into a set of simpler functions such that different functions must be performed when the inputs come from different regions of the input space. Although we consider the first form of task decomposition to be very important, our study focuses on the second form.

If a complex task is sufficiently well understood then an investigator can use domain knowledge to decompose this task into a set of simpler tasks. In this case, an advantage of our modular architecture is that the decomposition can be embedded in the architecture by individually training distinct networks to perform distinct tasks. Unfortunately, such extensive domain knowledge is rarely available. Therefore, our research emphasizes the need for a connectionist system that learns to perform task decomposition. Specifically, our objective is to develop a connectionist architecture

that learns when it is required to perform two or more functionally independent tasks and allocates distinct networks to learn each task.

The first major property of our architecture is that, by allowing networks to compete to learn the training patterns, this objective can be achieved. Consider training an architecture that consists of two networks, possibly with different topologies, to perform the function

$$f(x) = \begin{cases} x & \text{if } x < 0 \\ \sin x & \text{if } x \geq 0. \end{cases} \tag{1.1}$$

At each time step, both networks receive an input x and each computes an output. The network whose output is closest to the desired output wins the opportunity to learn about the given training pattern. The losing network does not adjust its weights. Using a slightly more sophisticated competitive mechanism, our architecture allocates its networks such that, after many time steps, one network learns to compute $f(x) = x$ for $x < 0$ and the other network learns to compute $f(x) = \sin x$ for $x \geq 0$. Thus, the architecture is said to perform function decomposition in the sense that it learns that the training patterns are generated by two distinct functions and allocates different networks to learn each function.

Suppose that in the above example, the architecture consists of one single–layer linear network and two multi–layer nonlinear networks. As is discussed in Chapter 2, the architecture may achieve the best generalization if it allocates the linear network to learn the linear portion of f and one of the nonlinear networks to learn the nonlinear portion of f. The second major property of our modular architecture is that it tends

to allocate to each function the network whose topology is most appropriate to that function. Once again, this ability results from the competition among networks to learn the training patterns.

Above we assumed that when a complex task is decomposed into a set of simpler tasks, then a different network should learn each of the simpler tasks. However, this strategy fails to take advantage of beneficial generalization across tasks. Several researchers (e.g., Selfridge, Sutton, and Barto [66], Jacobs [31]) have noted that training a system to compute one task may facilitate the system's ability to learn a second task, a situation referred to as positive transfer of training, or it may retard the system's ability to learn a second task, a situation referred to as negative transfer of training. The third major property of our modular architecture is that, in many circumstances, similar tasks are learned by the same network, resulting in the benefits of positive transfer of training, whereas dissimilar tasks are learned by different networks, thereby avoiding the detrimental effects of negative transfer of training.

The last major property of our modular architecture concerns the assumption that a complex task should be decomposed into a set of distinct simpler tasks. Minsky [48] noted that a disadvantage of modular systems is that they hinder the sharing of knowledge across modules. This problem arises when a complex task is best approached by utilizing a shared strategy in all contexts plus modifications to this strategy that are context dependent.

For example, consider the task of controlling a robot arm to move a variety of payloads, each of a different mass, along a specified trajectory. One decomposition of this task is to create separate controllers for different mass categories (e.g., light, medium, or heavy payloads). However, a better decomposition is to utilize a shared controller that controls the arm when it contains no payload along with other controllers, one for each mass category. The shared controller provides torques to the arm at all times. The other controllers add extra torques in a context sensitive manner. Thus, one controller adds extra torques when the object is light and a second controller adds extra torques when the object is heavy.

As is demonstrated in Chapter 5, a slight modification to our modular architecture allows it to allocate one network to learn to be the shared controller and the other networks to learn to be the controllers for the different mass categories. Therefore, the fourth major property of our architecture is that it can learn to solve a task by learning a shared strategy that is used in all contexts along with a set of modifications to this strategy that are applied in a context sensitive manner.

1.1 Outline of the Thesis

This thesis is organized as follows. Chapter 2 lists several advantages that suitably designed modular connectionist architectures have over single networks. Chapter 3 presents the modular architecture that we have developed and details the equations that govern the architecture's behavior. Chapter 4 reports the results of training

single networks and modular architectures on the "what" and "where" vision tasks studied by Rueckl, Cave, and Kosslyn [62]. Chapter 5 reports the results of training single networks and modular architectures to control a simulated robot arm to move a variety of payloads, each of a different mass, along a specified trajectory. Chapter 6 discusses the relationship between the structure of a modular architecture and the task decomposition discovered by the architecture. The last chapter, Chapter 7, draws final conclusions and suggests extensions to the research described here.

CHAPTER 2

ADVANTAGES OF MODULAR CONNECTIONIST ARCHITECTURES

Theorists have shown that connectionist networks are universal approximators, meaning that for any given function there is a connectionist network capable of approximating it arbitrarily closely (e.g., Hornik, Stinchcombe, and White [30]). Given this result, what advantages might a modular architecture consisting of several connectionist networks have over a single network? In this chapter we answer this question by arguing that modular architectures have advantages in terms of learning speed, generalization capabilities, representation capabilities, and their ability to satisfy constraints imposed by hardware limitations. This discussion is not specific to the architecture that we have developed; it applies to the general class of multi–network systems of which ours is an example.

2.1 Learning Speed

Several characteristics of modular architectures suggest that they should learn faster than single connectionist networks. One such characteristic is that modular architectures can take advantage of *function decomposition*. If there is a natural way to decompose a complex function into a set of simpler functions, then a modular architecture should be able to learn the set of simpler functions faster than a single network can learn the undecomposed complex function. For example, consider the absolute value function

$$f(x) = \begin{cases} -x & \text{if } x < 0 \\ x & \text{if } x \geq 0. \end{cases} \tag{2.1}$$

This nonlinear function can be learned by a single network with at least one layer of hidden units. Alternatively, it can be learned by a modular architecture consisting of two networks, each a single linear unit, and a mechanism for switching on the appropriate network in the appropriate context. One network can learn the function $f(x) = x$, and the other network can learn the function $f(x) = -x$. Assuming that it is relatively easy to learn the switching mechanism, the modular architecture should be able to learn faster than the single network because it does not use hidden units and each module only needs to learn a linear function.

In addition to being able to take advantage of function decomposition, modular architectures can be designed to reduce the presence of conflicting training information that tends to retard learning. We refer to conflicts in training information as

crosstalk and distinguish between spatial and temporal crosstalk. Spatial crosstalk occurs when the output units of a network provide conflicting error information to a hidden unit. Jordan [34] and Plaut and Hinton [57] noted that this occurs when the backpropagation algorithm is applied to a single network containing a hidden unit that projects to two or more output units. For example, suppose that a hidden unit projects via positive weights to two output units and that when compared to the desired output values, the output level of the first output unit is too small, whereas the output level of the second output unit is too large. Using the backpropagation algorithm, the first output unit provides derivative information specifying that the hidden unit should have a larger output. However, the second output unit provides derivative information specifying that this same hidden unit should have a smaller output. This conflict in derivative information is an instance of spatial crosstalk.[1]

Plaut and Hinton [57] noted that a modular architecture consisting of a separate network for each output unit is immune to spatial crosstalk. For example, consider the systems shown in Figure 2.1. Panel A shows a single network and Panel B shows a modular architecture consisting of three separate networks, one for each output unit. Although these systems can be applied to the same tasks, the modular architecture

[1]Although spatial crosstalk is clearly seen in terms of the backpropagation algorithm, it is not limited to networks trained using this algorithm. A network trained using any algorithm that approximates gradient descent (e.g., the A_{R-P} algorithm of Barto and Anandan [9] and Barto and Jordan [10]) can suffer from spatial crosstalk.

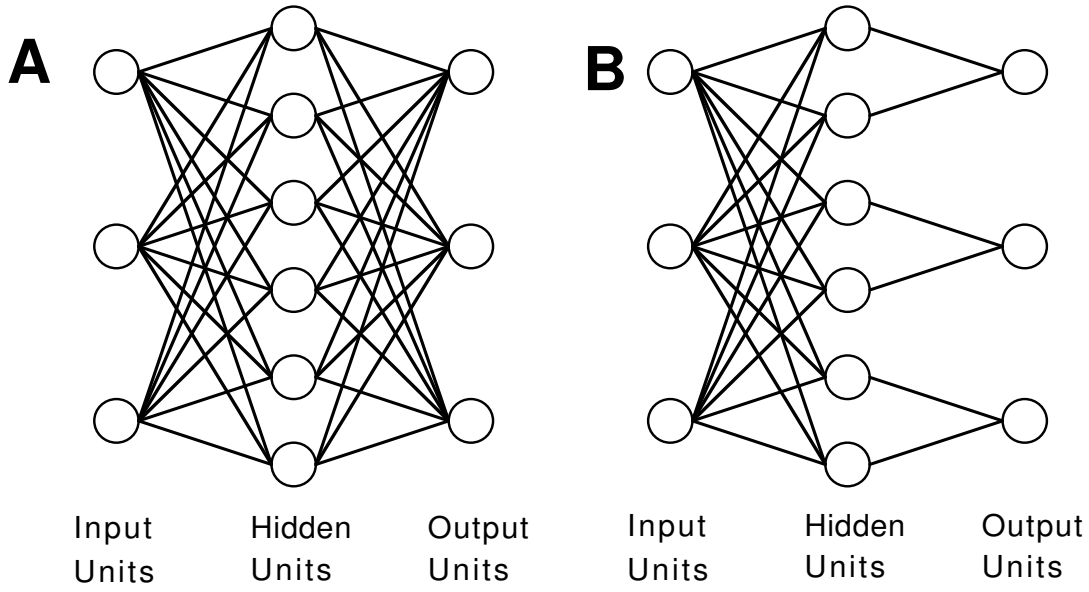

Figure 2.1: A: A network that is susceptible to spatial crosstalk. B: A modular architecture that is not susceptible to spatial crosstalk.

is immune to spatial crosstalk because each hidden unit projects to a single output unit.

In contrast to spatial crosstalk—where a unit receives inconsistent training information at a single instant in time—a unit might receive inconsistent training information at different times, a situation resulting in temporal crosstalk. One form of temporal crosstalk occurs when a network is trained to perform different functions at different times. For example, suppose that when a network is trained to perform one function, some of its hidden units become particularly useful in performing that function. When this same network is later trained to perform a second, different function,

one would like it to learn the second function without unnecessarily degrading its performance on the first function. This can be accomplished if the hidden units that participate in implementing the second function are not the ones that implement the first function. However, as Sutton [67] observed, gradient descent algorithms produce the opposite behavior because they preferentially modify the weights of the hidden units that are already useful. Therefore, after training on the second function, the network's performance on the first function will tend to be significantly degraded.

Temporal crosstalk arises in another situation closely related to that just described: when training patterns come from one region of the input space for many consecutive trials and later come from a different region for many consecutive trials. [2] If the function has different properties in these different regions, the network would receive conflicting training information. Training with patterns in the second region of the input space can result in degradation of the network's ability to perform correctly for patterns in the first region.

The conflicts in training information producing spatial and temporal crosstalk can be thought of in terms of transfer of training. Training a system to compute one function may facilitate the system's ability to learn a second function, a situation referred

[2] Although this is not how one would typically prefer to train a network, it may be unavoidable. For example, in training a network to control a dynamical system, the state of the dynamical system is likely to change slowly relative to the rate at which the system's variables are observed. Hence, the network is likely to receive inputs from the same region of the input space for many consecutive time steps.

to as positive transfer of training, or it may retard the system's ability to learn a second function, a situation referred to as negative transfer of training. Suitably designed modular architectures should learn faster than single networks because similar functions can be learned by the same network of the modular architecture, resulting in the benefits of positive transfer of training, whereas dissimilar functions can be learned by different networks, thereby avoiding the detrimental effects of negative transfer of training.

2.2 Generalization

Unless the structure of a network is well–matched to the function on which it is trained, it is unreasonable to expect the network to generalize well (e.g., Denker et al. [16]). For example, although both a single–layer linear network and a multi–layer nonlinear network can accurately learn a set of training pairs generated by a linear function, one would expect the single–layer network to generalize better because its structure is closer to the structure of the function being learned. If a modular architecture were able to decompose a complex function into a set of simpler functions and allocate an appropriately structured network to each simpler function, then one would expect good generalization. In this case, the mechanism responsible for allocating training patterns to the networks of the modular architecture would be automating part of the process of matching network structures to tasks, the other part of the process being played by the initial choice of the modular architecture's repertoire of networks.

A second reason that a modular architecture should generalize better than a single network involves the difference between local and global generalization. Global generalization occurs when the learning of a training pattern from one region of the input space influences the network's performance on patterns from a much wider region of the input space. As suggested above in the discussion of temporal crosstalk, when the function generating the training patterns possesses different characteristics in different regions, global generalization is an undesirable property due to negative transfer of training. In contrast, modular architectures perform local generalization in the sense that each network of the architecture only learns patterns from a limited region of the input space. Therefore, training a modular architecture on a training pattern from one of these regions does not affect the architecture's performance on patterns from the other regions.

2.3 Representation

Modular architectures tend to develop representations that are more easily interpreted than the representations developed by single networks. By this we mean that it tends to be easier to understand how a modular architecture implements a function than it is to understand how a single network implements the same function. This property was demonstrated by Rueckl et al. [62] who trained two connectionist systems to perform object recognition and spatial localization from simulated retinal images. As a result of learning, the hidden units of the system using separate networks

for the recognition and localization tasks contribute to the solutions of these tasks in more understandable ways than the hidden units of the single network applied to both tasks. In the former system, a different set of hidden units is used to represent information about the different tasks. In the latter system, on the other hand, the same set of hidden units is forced to represent information about both tasks despite the fact that the two tasks are relatively independent.

In addition to making it easier for experimenters to understand their systems, interpretable representations may make it easier for one portion of a system to understand the tasks performed by other portions. Minsky [48] argued that if one module can determine the task that is performed by a second module, then the former module can request the latter module to perform that task in the appropriate situation. This view emphasizes that the functions learned by the modules can be thought of as "building blocks" to be used on other occasions in the performance of more complex tasks.

Another advantage of using different networks to perform different portions of a task is that this representation facilitates the use of attentional mechanisms. Cowey [14] speculated that properties of a visual image may be attended to or unattended to through the operation of a relatively coarse mechanism that enhances or suppresses the activations of entire populations of processing units. Only if different modules of an architecture represent different portions of a task would we generally expect such mechanisms to have coherent and sensible effects.

Other reasons that modular architectures are superior to single networks in their representational capabilities are related to familiar arguments advanced in the computer science and psychology literatures: because modular architectures localize functions and develop more interpretable representations, they are easier for researchers to debug; because much of human knowledge is modularized, it is easier for researchers to embed domain knowledge into a connectionist system when the system is organized in a modular fashion; and because modules can be added one at a time, modular architectures facilitate the development of connectionist systems in an incremental manner.

2.4 Hardware Constraints

Another property of modular architectures suggesting advantages over single networks is that modular architectures can more closely satisfy several types of constraints imposed at the level of hardware implementation. In particular, suitably designed modular architectures can reduce the number of units and the lengths of connections.

In a discussion of representations employed by the brain, Ballard [5] suggested that a limitation on the number of neurons compels the brain to adopt a modular architecture. He hypothesized that the brain uses a coarse–code (Albus [1], Hinton [28]) to represent multi–dimensional spaces. Using this kind of representation, the number of neurons required to represent a space is $\frac{N^k}{D^{k-1}}$, where k is the dimension of the space,

N is the number of just–noticeable differences in each dimension, and D is the diameter of the receptive field of each neuron (Hinton [28], Ballard [5]). Because this rapid growth in the number of neurons limits the number of dimensions that can be represented in a cortical area, high–dimensional spaces must be represented in such a way that different dimensions are represented in different areas. Because different areas represent different dimensions, these areas must compute different functions. Analogously, in order to reduce the number of units required by a connectionist system, one may distribute the representation of multi–dimensional spaces among multiple networks.

Other speculations as to why the brain contains multiple cortical areas are also relevant to connectionist systems. Cowey [13] suggested that if the cortex uses lateral inhibition to sharpen various visual attributes, such as edges, orientation, color, disparity, spatial frequency, size, and movement, then a retinotopic representation of these attributes allows the use of relatively short connections between neurons since this representation places the interneurons necessary for receptive–field tuning relatively close together (Kohonen [39], Durbin and Mitchison [18]). If neurons for all the different attributes are represented in a single retinotopic map, however, the connections needed to sharpen the tuning of individual neurons for one of these attributes would be unnecessarily long. The existence of multiple retinotopic maps, each in a separate cortical area, allows the cortex to highlight several attributes using short local connections.

Similarly, Barlow [7] suggested that in order to detect various visual attributes using neurons whose connections are of minimal length, the cortex must employ multiple representations. He argued that because the brain primarily contains local connections, it is difficult, using a retinotopic representation, to detect similarities among non–contiguous locations of the visual field. In order to detect such similarities, it is necessary to map the information in the retinal image so that similar events are represented close to each other independently of the retinal coordinates of the events. For example, using a retinotopic representation and neurons with local connections, it is difficult to detect the co–linearity of line segments located at different places in the retinal image. However, in a different, non–retinotopic representation (e.g., the Hough transform (Duda and Hart [17], Ballard [4])) nearly co–linear line segments can be represented by neighboring neurons. In general, the existence of multiple representations, each in a separate area, allows the brain to detect several attributes using short connections.

The hypotheses of Ballard [5], Cowey [13], and Barlow [7] outlined above about why the cortex employs multiple representations located in different cortical areas are also relevant to connectionist architectures. These hypotheses suggest that in order to evaluate several attributes using units with local connections, connectionist systems should employ multiple representations located in different networks.

This completes our discussion of the advantages of modular architectures in relation to single networks. Based on this discussion, one might be tempted to conclude that it is best to use a modular architecture consisting of networks that each compute

a highly restricted domain–specific function. However, as noted by Fodor [22], certain portions of the human mind appear to be modular (for our purposes, we can take Fodor to mean that they compute domain-specific functions) whereas other portions seem to be non–modular. This suggests that there may be advantages to a connectionist architecture that includes both networks that compute fairly restricted domain–specific functions and networks that compute more general domain–independent functions. In addition, Marr [43] emphasized that modularity (in terms of computing domain–specific versus domain–independent functions) is a continuous and not a binary attribute. To this, we add that a network that learns a function in the context of one domain may later be found useful in other domains. Thus, a network that at first is thought to compute a domain–specific function may, due to transfer of training, later be thought to compute a domain–independent function.