The Development of a Research Environment for Neural Networks: Instantiating Neocognitrons

Andrew J. Czuchry, Jr.

Technical Report: GTRI/AIB/TR:901221

The Development of a Research Environment for Neural Networks: Instantiating Neocognitrons

Andrew J. Czuchry, Jr.

Technical Report: GTRI/AIB/TR:901221

Contents

List of Figures

The Development of a Research Environment for Neural Networks: Instantiating Neocognitrons

Andrew J. Czuchry, Jr.
Georgia Institute of Technology
Artificial Intelligence Branch
Georgia Tech Research Institute
Atlanta, GA 30332

December 21, 1990

Abstract

Neural networks can be thought of as combinations of generic pieces linked together in varying architectures. Many different models and architectures have been presented in the published literature. Networks may differ both in the characterization of their pieces and in the connection patterns of those pieces. In order to exploit the similarities between models, incorporate the differences between models, and automate the process of linking the pieces together, a prototype of a generalized research environment for neural networks is being developed. The main virtue of this generalized environment is the flexibility it provides for testing various neural network architectural and processing decisions without having to write programs. The environment encompasses the ability to specify desired characteristics (e.g., activation functions, connection masks, sub-net sizes) as parameters to network creation functions; it does not force a programmer to combine such characteristics by altering the program code itself. The specification of characteristics and the resulting automatic creation of the corresponding neural network is herein referred to as the instantiation of a neural network. Alternatively, this process can be thought of as the dynamic creation of neural networks. Dynamic creation is achieved by the computation of connection patterns and node organizations *within* the environment; computation is performed by generic creation routines, not in user written routines. Standard modularization techniques have also been used to facilitate activation rule and/or learning rule modification (a priori). The viability of this research environment has been demonstrated by its use in the development of a generalized implementation of Kunihiko Fukushima's neocognitron. This paper initially introduces the generalized research environment, subsequently discusses the architecture of a test case network (the neocognitron), and finally presents the initial results in testing a neocognitron instantiated by the environment.

1 Introduction

Testing neural network architectures can be a tedious process. Typically, a researcher utilizes a collection of generic routines, either purchased or developed in-house, for computing and displaying input and output activations of connected neurons. One of the main problems with this situation is that in order to test varying architectures, new routines must be written which combine the network nodes in the appropriate connection architecture and incorporate the respective activation functions. The new routines for setting up the architecture must subsequently be debugged themselves before the complete architecture itself can be tested. Emphasis on empirical architecture analysis is typically not considered as a component of the generic package. In order to ameliorate the myriad of foreseen complications in empirically analyzing numerous and varying architectures for pattern recognition using digitized images, a project has been undertaken to develop a generalized neural network research environment with the flexibility to encompass arbitrary network architectures. Additionally, the environment has been designed to dynamically put together the pieces of an intended network architecture in much the same way a programmer would. The key advancement over more conventional approaches is that the pieces are woven together automatically within the environment.

Given the challenge of creating a generalized neural network research environment, development began with the necessary primitive components of neural networks: 1. activation functions, 2. learning rules, and 3. connection and connection weight representations. Subsequently, routines were designed which put together these primitive components based upon parametric specifications. For example, procedures were written to *calculate* connection patterns based upon network characteristics, such as the size of component networks (i.e.,

projection areas) and the types of nodes within each component network (i.e., projection specification, as in 3 X 3 squares projecting from each node). The test domain for the environment has been Fukushima's neocognitron (Fukushima, 1980). At present, any neocognitron can be instantiated (i.e., any combination of plane sizes, number of planes per layer, and number of layers) by merely passing parameters to the network creation functions which calculate connection patterns, plane organizations, and layer hierarchies.

The creation of this environment has proceeded along two conceptual fronts. In order to achieve the goal of applying the neocognitron's architecture to pattern recognition tasks involving real-world images, an environment was developed within which neocognitrons with varying organizations could be dynamically created. Alternatively, there was a constant concern for the general purpose nature of the environment. The creation of the environment began with the development of generalized knowledge/data structures incorporating the requirements of the neocognitron neural network model in conjunction with being extensible to alternative neural network models as well. Additionally, as indicated above, routines were developed to calculate connection patterns based upon desired characteristics of the network. Finally, a collection of display routines has been written and subsequently used to debug and analyze the systems created by the environment.

The combination of generalized knowledge/data structures, connection computation, and analysis displays has been evolved into an environment which can be tailored to function as a general purpose and extensible neocognitron. The general purpose nature and extensibility are direct results of the dynamic creation of the connection architectures and hierarchical network dimensions based upon the parameterized calculations utilizing the generalized knowledge/data structures. The belief is maintained that the general-purpose nature of the design was a valuable contribution to this project because it does not slow the actual process-

ing of the network once created, yet it does provide for flexible extension of the architecture of the neocognitron, or alternative architecture, as the instantiated networks are tested on real-world images. The development of the generalized knowledge/data structures and the initial character recognition testing has been successfully completed. The research effort is continuing with extensive testing on real-world images.

This paper describes the research environment and network instantiation. Although the research environment has been designed to instantiate arbitrary network architectures, the class of networks called neocognitrons is the sole specific network architecture discussed in this paper. The neocognitron class has been chosen as the example for this discussion because it provides a rich variety of issues to be addressed without cluttering the mind of the reader with the details and idiosyncrasies of numerous alternative model types. The instantiation process and the associated generic knowledge structures are fundamental to the general purpose nature of the research environment. The general discussion, therefore, applies to *any* network architecture. The detailed example itself applies specifically to the neocognitron class, yet it still elucidates the key issues of the instantiation of any architecture. As discussed in the next section of this paper, alternative architectures are produced by merely selecting different parametric specifications for the instantiation process. Clarification of these general ideas is achieved, in subsequent sections, through a specific example in the form of the neocognitron. In these latter sections, the neocognitron is reviewed and the challenges faced in designing the overall environment are highlighted. Finally, test results are presented and future directions are indicated. The test results are presented using the instantiation of a specific neocognitron as an example so that the accuracy of the instantiation process can be verified by comparison to other results available in the published literature. The general purpose nature of the fundamentals of the research environment (i.e., instantia-

tion and generic knowledge representation structures) will be further verified in subsequent papers demonstrating alternative network architectures. The emphasis of this paper is on the ideas behind the design and the architecture of the research environment. Details of the implementation are not significant for this presentation and are, thus, not presented in this paper.

2 The Research Environment

Neural networks have three main components: 1. processing elements which perform local computation, 2. connection architectures which tie processing elements together, and 3. learning rules which adapt the network. Rather than limiting the environment to a specific version of a particular network model, standard software modularization and artificial intelligence representation techniques have been utilized. Each of the network components has a modular organization and all of the information required for elemental computation is stored locally. The modular structure and localized representation of computation within the environment form the basis for the realization of a general-purpose research environment for neural networks. Many of the software techniques applied in the development of this environment are not new, yet their combined effect has produced a valuable research tool and many significant insights into how such a system would be designed for general distribution.

Processing elements are structures characterized by activation functions and output functions. Many network models utilize processing elements with a summation activation function and a nonlinear output function (e.g., perceptrons (Rosenblatt, 1962). Other models utilize processing elements which have activation functions that incorporate multiplicative

combinations of the input elements and use various forms of nonlinear output functions (e.g., on-center off-surround shunting networks (Grossberg, 1973), Sigma-Pi units (Rumelhart et al., 1986b), and other Higher Order Networks (Lee et al., 1986)). Any activation function and output function can be incorporated into the processing elements utilized in the research environment. In order to handle such versatility, the appropriate activation functions and output functions are encoded within the processing element structure. References to the particular functions are used, rather than the functions themselves, to preserve modularity and extensibility of the environment. The functions themselves are distinct modules which are tied to the processing elements via the reference encoded in the processing element.

The connection architecture is comprised of two components: 1. connections and 2. connection weights. Connections tie processing elements together into a network. Connection weights bias the significance of the individual connections. With regard to the connections themselves, there are both output and corresponding input connections. The output connections indicate the other elements to which an element is connected. The input connections are structures that indicate which elements send their outputs to the element under consideration. Rather than store connection representations in global processing routines which must encode the entire connection architecture, the research environment is structured so that the connections for each processing unit are encoded within the processing element structure itself. Each processing element is, therefore, a self-contained unit independent of global routines. Each processing element has a representation of its output connections; the traversal of connections in the update process is based upon these locally encoded connections, not on a global representation for the connection architecture.

The connections are computed at the time of network instantiation and stored locally within

each element. The instantiation routines are structured such that the appropriate connections are computed based upon parameters passed to the respective routines. For example, if each element in a 5 X 5 network is to project into a 3 X 4 network such that each element has a 2 X 2 projection and all of the 3 X 4 network elements must be covered, then the system would compute that the element in position (0, 0) of the 5 X 5 network would be connected to the elements in positions (0, 0), (0, 1), (1, 0), and (1, 1) of the 3 X 4 network (see Figure 1). The element in position (4, 4) of the 5 X 5 network would be connected to the elements in positions (1, 2), (1, 3), (2, 2), and (2, 3) of the 3 X 4 network. A variety of standard connection architectures have been presented in the neural network literature (e.g., ART (Carpenter and Grossberg, 1987b; Carpenter and Grossberg, 1987a; Grossberg, 1976a; Grossberg, 1976b), back propagation (Rumelhart et al., 1986a), Hopfield networks (Hopfield, 1982), and the neocognitron (Fukushima, 1980)). Because the connection calculation routines are parameterized and actually calculate the connection patterns, arbitrary algorithmically expressed connection patterns can be realized.

The weight associated with each connection is encoded in a structure separate from the connection itself and stored in the processing element. The separation of the connections and the weights provides a decoupling which enables connection architectures to be varied independently of weight assignment/adaptation and vice versa. The connection information is available to the weight assignment routines. This information is provided to the assignment routines because the number of connections emanating from an element is an essential quantity in determining weight assignment, e.g., for a normalized uniform distribution. However, changing from a normalized uniform to an exponential weight distribution should not require re-calculation of the connection architecture, it should only require re-calculation of the weights. Likewise, changing the connection architecture should not alter the weight cal-

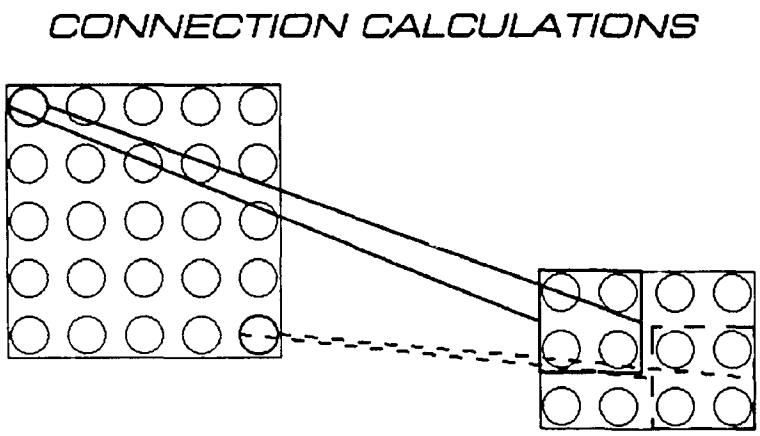

Figure 1: Connections calculated for 2 X 2 projections from a 5 X 5 to a 3 X 4 network.

culation *method*. The separation of the connection and weight representation provides the necessary flexibility for network design. The separation of the connections and the weights also provides for flexibility in the maintenance of both connections and weight values (e.g., during learning).

The learning rules applied in the field of neural networks are typically rules for adjusting the connection weights. There are a variety of different connection weight learning rules (e.g., Hebbian learning (Hebb, 1949), Local Interaction (Alkon, 1989; Alkon and Rasmussen, 1988; Alkon, 1984), Competitive Learning (Grossberg, 1987)). Within the research environment, the learning rules have been associated with the networks via reference to the appropriate procedures. This standard modularization technique provides a representation for each network to encode the appropriate learning rules. Changing the learning rules, thus, becomes a parametric change rather than a recoding effort. The learning rule routines are passed the respective network structures and perform calculation based upon those structures. As a

result, the learning rules naturally adapt to changing network organizations. The modularization of the learning rules ensures the parametric nature of the network instantiation and processing routines.

Through the process of actually using the research environment to instantiate neural networks and analyze their behavior, many interface needs arose. These needs formed the specification for the user interface and the user interface has been tailored to those needs. A by-product of the use of the environment is the recognition of various challenges in understanding neural network processing in visual terms rather than purely mathematical terms. These challenges have provided avenues for the investigation of interface properties necessary for general-purpose neural network environments. Initial results have shown full-color graphics displays to be a fundamental component of such visualization.

In keeping with the goal of developing a flexible, general-purpose neural network research environment, the display routines utilize the information contained in the network representation to tailor the displays. Thus, as the network and underlying models change, the displays will automatically be adjusted. Initial investigations have been quite fruitful. A collection of output displays is presented in the results section (Section 5). For ease of publication, the color displays have been reproduced as black-and-white images.

In summary, each component of the network 'carries around' all the information it needs for performing its specified functions. The functions themselves are separated from the representation encoded in each component in order to enhance and ensure modularity. The modularity is essential for the general purpose nature of the environment. In addition to the modularity of the networks instantiated within the research environment, a variety of utilities have been included for the developer/applications programmer/user of the environment. Full

color graphics displays are utilized to aid comprehension of the distributed processing within the network. Inputs are presented graphically and can be selected through a point-and-click window interface. The utilities which are provided have been chosen to aid development and analysis of neural networks and have been derived from the actual needs occurring in such processes.

3 The Neocognitron

The neocognitron is a hierarchical neural network model designed to perform visual pattern recognition tasks (Fukushima, 1980). The main virtue of the neocognitron's architecture is that it ameliorates the common problem of distorted or noisy input for pattern recognition. More specifically, the neocognitron is a pattern recognition architecture which is de-sensitized to both of two fundamental pattern recognition limitations: 1. positional shifts, and 2. pattern deformation (e.g., deformation due to noise or other forms of distortion).

The organization of the architecture and the basic processing elements of the neocognitron is fundamental to its functionality. The neocognitron is a hierarchical arrangement of multiple layers of sub-networks of artificial neurons. The neurons are often called cells in the context of the neocognitron. The cells are organized into primitive sub-networks which are called planes. The planes are organized into layers. The layers are the components with which a hierarchy is established. Each layer in the hierarchy is composed of planes (sub-networks) which contain both excitatory and inhibitory neurons. There are two main types of excitatory neurons, each organized into groups of planes: 1. excitatory variable connection weight neurons, S-cells, and 2. excitatory fixed and nonmodifiable connection neurons, C-cells. There are also

two types of inhibitory cells: 1. V_s-cells, which are variable connection weight inhibitory cells, and 2. V_c-cells, which are fixed and nonmodifiable connection network inhibitory cells. These inhibitory cells are used to ensure that the *absence* of certain features can be detected. They would fire and inhibit the excitatory cells (S-cells and C-cells) from firing if the features to which the inhibitory cells are sensitive are present. Figure 2 depicts the generalized connection architecture between cells in layers. For simplification of the figures, however, V-cells have not been presented. In fact, only representative cells are presented for the excitatory S and C cells. The layout of Figure 2 is significant for this paper because it will be used in presenting the results. Therefore, the layout will initially be described and subsequently the connection patterns will be presented.

The six processing layers of the neocognitron's hierarchy are depicted as double rows in Figure 2. The input layer is depicted in its own row at the bottom of the figure. Each layer is labeled according to the characteristics of the planes (i.e., sub-networks) contained in that layer. In other words, the layer S_1 contains sub-networks of modifiable S cells; the layer C_1 contains sub-networks of nonmodifiable C cells. The number of cells per plane decreases as one moves from the input layer to the C_3 'output' layer. The final C layer contains planes of *single* cells. These single cells are used for recognition. The individual cells do not have a priori recognition significance, but through the course of training, they come to signify a particular class of patterns. This concept will be further elucidated in the results section (Section 5).

The cylindrical projection patterns depicted in Figure 2 represent the connection patterns of particular cells in a layer. The dotted projections from layer S_1 to the input layer all emanate from a cell in the same position in the corresponding S plane; each projects to the same position in the preceding layer. The mid-length dash projections from layer S_1 to

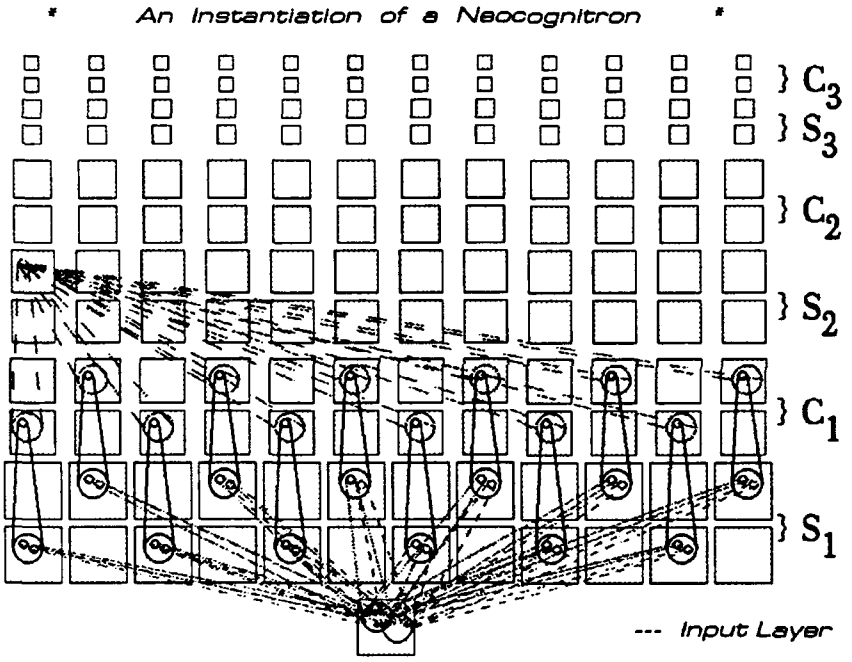

Figure 2: Connections between layers of the hierarchy in a neocognitron.

the input layer all emanate from a cell in the same position in the corresponding S plane. The distinction between mid-length dash and dotted projections is that they are in different positions within each S - plane and, therefore, project to different positions in the preceding layer (input layer). Notice that each projection occupies approximately one-quarter of the input plane. Each dotted projection is looking for a different pattern in the same position in the preceding layer. Each mid-length dash projection is looking for the same pattern as the dotted projection from the same plane, yet it is looking in a different position. This is the initial foundation for the noise elimination characteristics of the neocognitron; multiple cells in each plane are looking for the same pattern in different positions and the cells in each plane are looking for different patterns in the preceding layer.

The second piece in the noise smoothing process is depicted in the solid projections from the C_1 planes to the S_1 planes. These solid projections represent the *position assimilation* connections of the C-cells. Active cells within the solid projection areas are treated nearly equally. In other words, if a particular mid-length dash cell is active rather than the corresponding dotted cell, it can make little difference to the C layer cell. It can become active regardless of the position of the activation pattern within it's solid projection area. This is further exemplified in Figure 3. Before discussing the interaction, however, one last point needs to be made with regard to Figure 2. The long dash projections from the S_2 cells to the C_1 cells demonstrate that each cell projects to cells in the same position over *all* sub-networks (planes) in the preceding layer. (Note that only a representative collection of projections is displayed in order to keep the figure from becoming too cluttered.) Additionally, one can see from Figure 2 that the long dash cells of layer S_2 are sensitive to the characteristics of nearly half of the original input layer (both dotted and mid-length dash areas) because of the position assimilation and noise smoothing of the C_1 layer. The cells in the final (C_3)

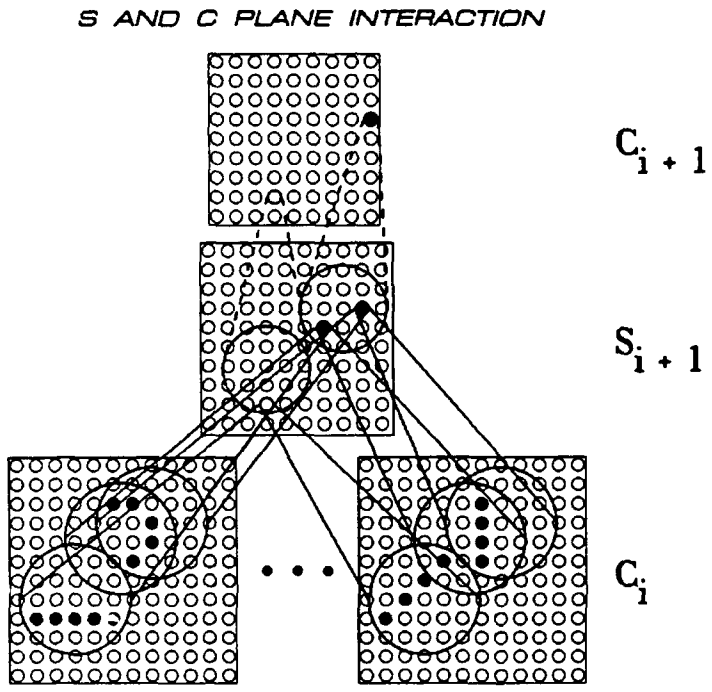

Figure 3: Interaction between cells in the C and S layers of a neocognitron.

layer are sensitive to the entire original input layer.

A detailed depiction of the interaction between S and C cells is presented in Figure 3. Each cell in an S_{i+1} plane is 'looking' for the same thing, but in a different position. That is, each has the same weight vector but a different position vector. This is demonstrated by the two active cells (filled-in circles in the S_{i+1} plane) which receive the appropriate hook and vertical line, whereas the horizontal and diagonal lines do not excite the cell which receives them as input. The particular input characteristics, to which the cells in a plane are sensitive, are

determined through training.

The position assimilation characteristics of the cells in the C plane are also represented in Figure 3. A cell in the C_{i+1} layer becomes active whenever the cells of its input projection area are sufficiently active. This property is termed ***position assimilation.*** The effect of position assimilation is that the C cells smooth noise, distortion, and positional shifts. The final piece of the noise amelioration puzzle is, thus, the combination of the stepwise assimilation of position and the extraction of features in a hierarchical manner. Noise is not removed all at once. Rather noise is removed little by little in each layer of the hierarchy with the final result being an ability to handle a significant amount of noise.

The criteria for the dynamic creation of this model is based upon the work presented by Fukushima in the published literature. The primary reference was (Fukushima and Miyake, 1982). Two additional publications, (Fukushima, 1980) and (Fukushima, 1988), were consulted as necessary. While the formulae and descriptions presented in Fukushima's writings form the basis for the effort, an entire environment has been created which can be used to dynamically create and test a variety of architectural alternatives implied by the variance in the characteristics of the neocognitrons Fukushima has developed. An additional distinction in this work is the intended test domain. The test domain for Fukushima's work has been character recognition. The goal of the present research effort is to test the effectiveness of the neocognitron in pattern recognition tasks applied to real-world images.

4 Challenges Faced

A significant number of challenges faced in the development of the research environment were particular to the initial test case model, the neocognitron. The foremost challenge in developing instantiations of the neocognitron (Fukushima and Miyake, 1982) was the sheer magnitude of the problem. There are a total of over $2.3M$ connections, each with it's own weighting factor, between the $10K$ cells and 145 planes in the network. Additionally, there are several parameters which interact and affect the behavior of the network. For example, each of the seven layers (excluding the input layer) has an intensity of inhibition parameter to control the amount of noise tolerated in matching a pattern; this parameter interacts with both the excitatory and inhibitory weights as an output is computed for a particular network cell. To date, the general behavior of varying cell excitation and single final layer cell activation contingent upon pattern presentation and prior training, as presented by Fukushima (e.g., (Fukushima and Miyake, 1982)), has been reproduced. With regard to interaction of changes in the parameters 'provided' by Fukushima, minor changes in any of the values (e.g., initial connection weights, inhibition control parameter, rate of reinforcement during learning) significantly affected the final behavior of the system. Since there was no specific direction for adjusting parameters, a generate-and-test paradigm was used until results became acceptable.

Other challenges faced were issues concerning the research environment itself. One such challenge was the development of a *usable* user interface and system utilities. The reason a usable user interface was needed is that interface requirements were the same as the needs for debugging, in the most efficient manner possible, the inadequacies of various choices for network parameters and architectures. In fact, the debugging needs formed the specification

for the interface. System utilities for writing and reading trained networks were also required and developed. All of these necessities added to the complexity and effort required in the completion of the project.

As a final comment with regard to neural networks in general, a heathly respect and skepticism for the currently proclaimed power and versatility of such systems has been gained. Although it has previously been realized that there are significant limitations in the current neural computing technology, the way in which these limitations can be corrected is certainly non-trivial. In fact, there is a question of the ultimate cause of these limitations; the cause is not intuitively obvious and remains to be discovered. The frustration of trying to debug a distributed representation, where it is not at all clear wherein the problem lies, was initially quite discouraging. Mathematical analyses can be quite tedious. Empirical adjustment can limit the overall power of the resulting network. Further consideration of the problems, however, provided a spark of enthusiasm for continuing to search for a resolution.

5 Results

The results presented in this paper demonstrate the viability of the research environment and indicate the accuracy of its instantiation process. In order to have a solid basis for this accuracy claim, the results of testing a standard neural network architectures are shown. The test case network is a neocognitron. The neocognitron has been chosen because of the large size of the network and the complexity of its connection architecture. The results presented in this section are the actual response characteristics of the instantiated neocognitron with regard to a variety of input patterns both before and after training. Additionally, a

collection of test patterns which the instantiated network correctly recognizes is presented. The accuracy of the instantiated network (neocognitron) can be verified by a comparison of these results to the results presented in (Fukushima and Miyake, 1982). Such a comparison indicates that the results presented herein demonstrate that several significant successes have been achieved with regard to the neocognitrons instantiated by the environment:

- Different patterns produce different excitation patterns within the network.
- Training of the network alters the excitation patterns of the network.
- After training, only a single cell fires at the recognition layer in response to different stimulus patterns.
- Appropriate clusterings are achieved for multiple versions of various numeric characters.

Before training, the activation of the cells within the network is heavily influenced by the characteristics of the input pattern. Input of a "3" produces activation quite different from the input of a "2" or a noisy "2". This is depicted in Figures 4, 5, and 6. Note that in the C_1 layer, the activation patterns for the different input patterns are all quite unique. This is especially significant for the "2" and "noisy 2" since they should be clustered in the same class, not in distinct classes. Through the process of training, the "2" and "noisy 2" will be clustered together. One additional comment should be made with regard to the initial activations before training. Notice that no activation is present in the cells of the $S_2 \rightarrow C_3$ layers. The reason is that the S layer activations are a function of the *differences* between the excitation patterns of each plane in the preceding layer. The differences are produced by the particular weighting function chosen within each projection pattern. In other words, the differences are realized because the connections between the S_{i+1} and C_i layers and the

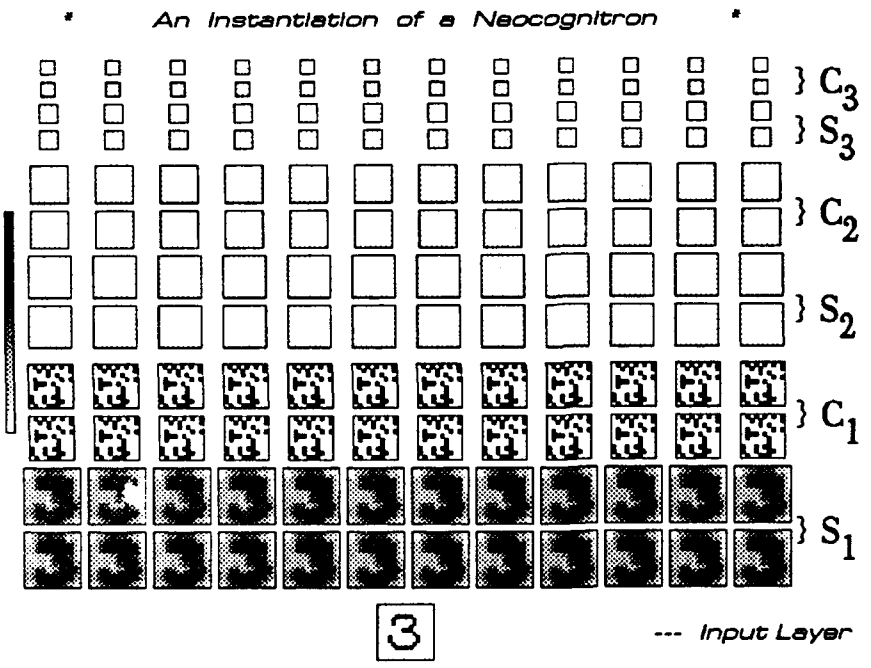

Figure 4: Activation pattern, before training, in response to the input pattern of a 3.

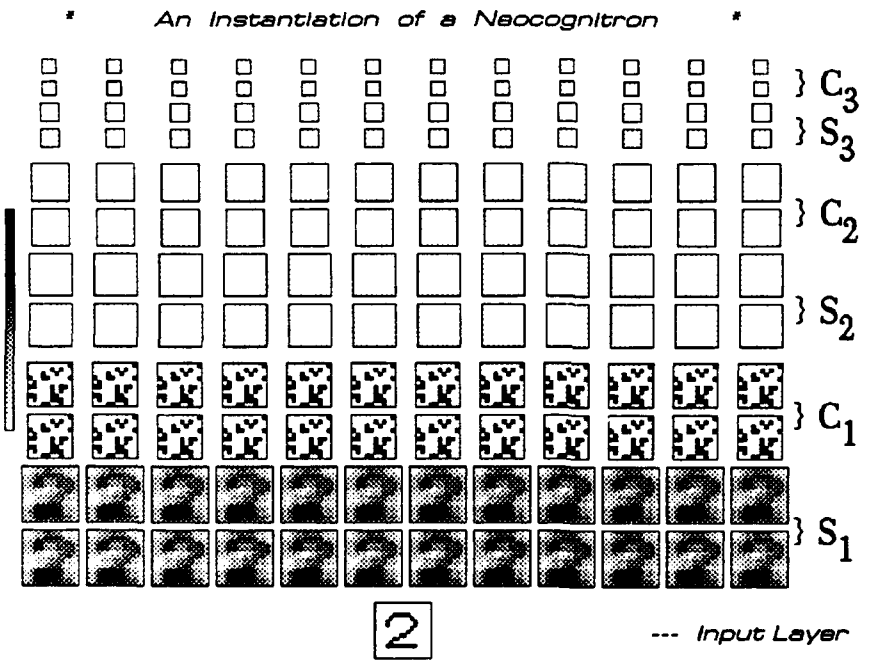

Figure 5: Activation pattern, before training, in response to the input pattern of a 2.

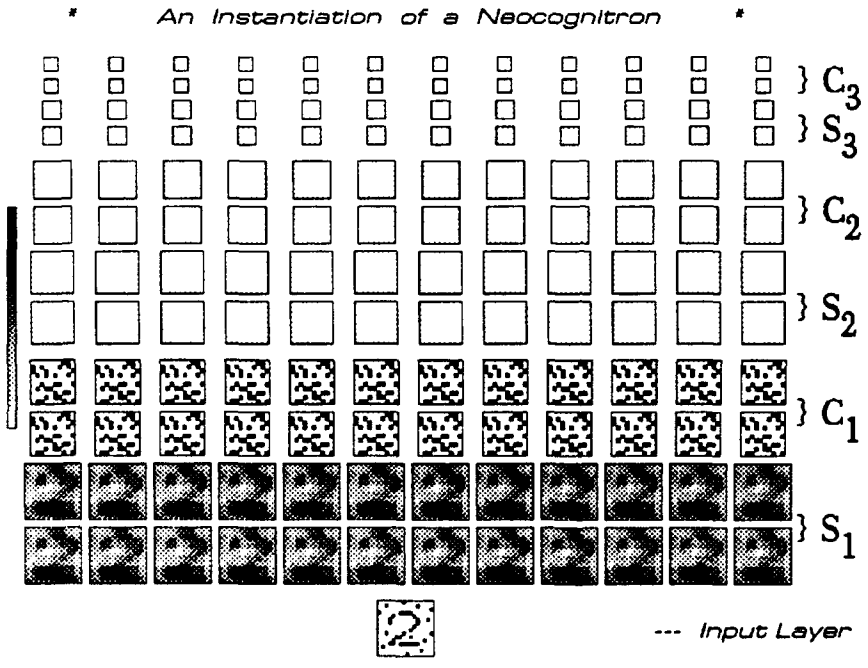

Figure 6: Activation pattern, before training, in response to the input of a noisy 2.

connections between the C_i and S_{i-1} layers are weighted by some distribution function and the activations are computed as a function of the weighting function. For the instantiation presented in this paper, the weighting function for each plane has been chosen to be a uniform distribution. Therefore, each plane within the C_1 layer (and also the S_1 layer) has an identical activation pattern. The cells in the S_2 layer are inactive due to the uniform connections to identical patterns (i.e., no differences are detected). This characteristic only affects the initial biases as it is changed through training. After training, the connections are weighted differently and activation is ultimately propagated all the way to the C_3 (output) layer.

The activation patterns within the instantiated neocognitron *after* training is presented in Figures 7, 8, and 9. The same input patterns are used as in the *before* training example, yet the activation patterns in layers $S_1 \rightarrow C_3$ are quite different before and after training. Two important points should be acknowledged. The first point is that cells in each plane have become sensitive to different characteristics (features) of the input pattern and, thus, produce activation patterns that are different in each plane within a layer. The second point is that distinct C_3 cells are most active for distinct classes, whereas the same C_3 cell is most active for common classes even if distortions of the input pattern occur. Note that in Figure 7, the system has *arbitrarily* chosen the cell in the C_3 layer which responds for a particular pattern. The important point is not which cell in particular responds but rather that the *same* cell at the C_3 layer responds to the corresponding input pattern of the same class regardless of positional shifts or pattern deformation. For example, Figures 7 and 8 have different most active cells in the C_3 layer because Figure 7 has a "3" as the input and Figure 8 has a "2" as the input. However, Figures 8 and 9 have the same most active cells because each represents the activation of the network in response to a variation of a "2"

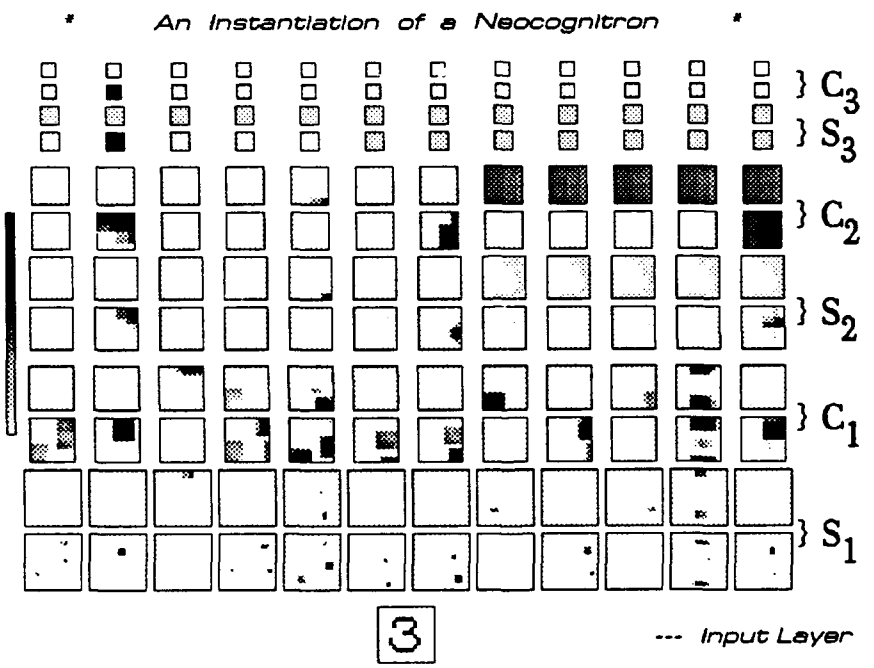

Figure 7: Activation pattern, after training, in response to the input pattern of a 3.

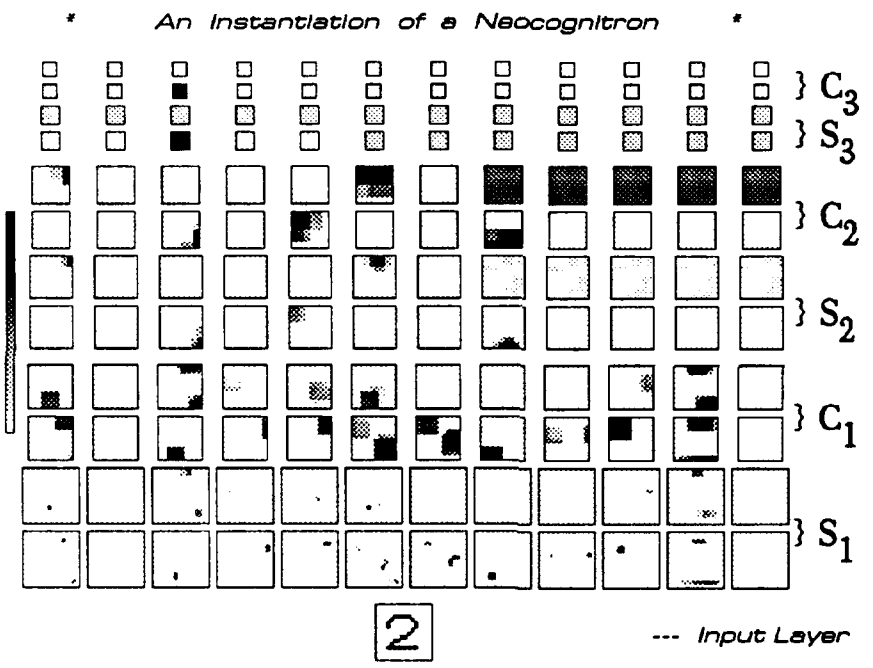

Figure 8: Activation pattern, after training, in response to the input pattern of a 2.

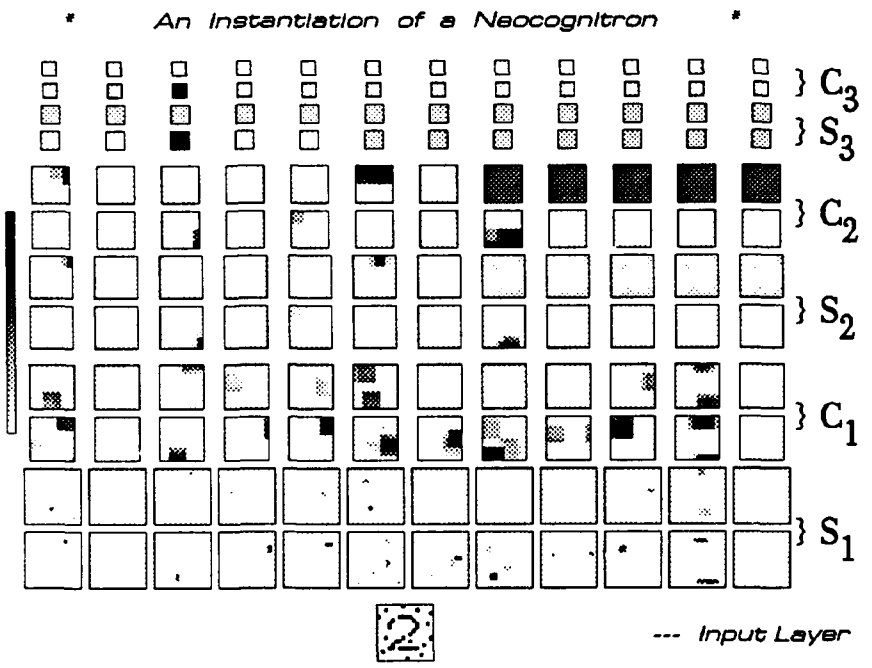

Figure 9: Activation pattern, after training, in response to the input of a noisy 2.

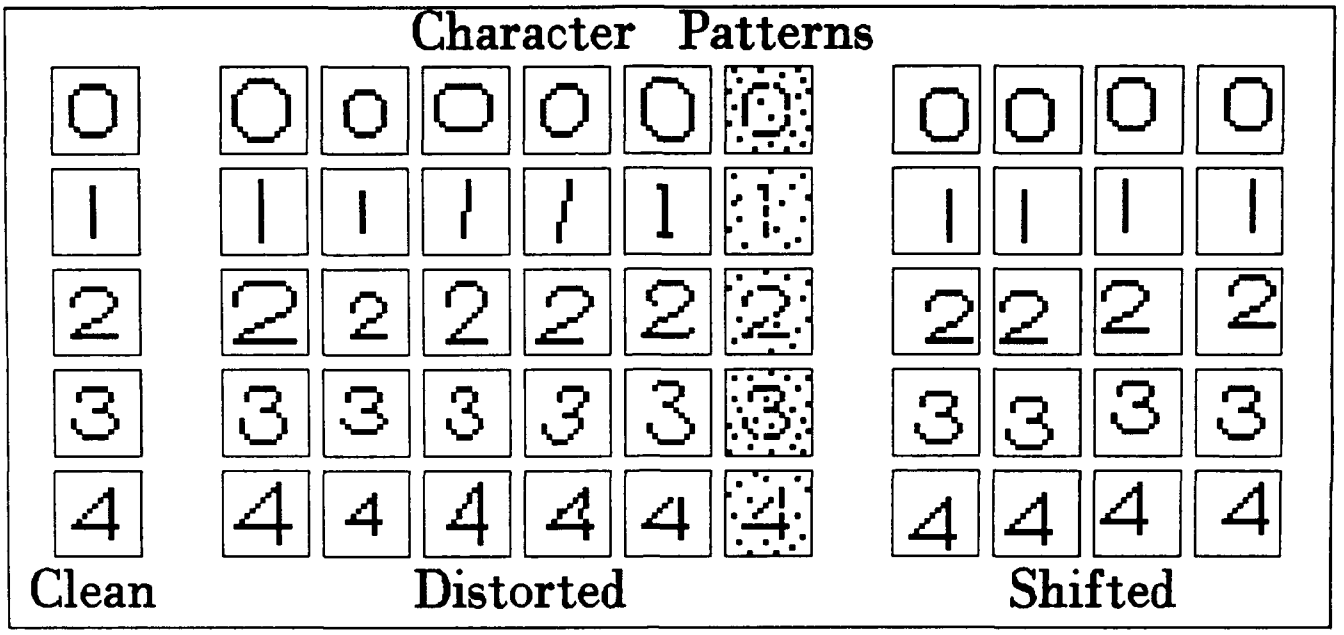

Figure 10: Some examples of the stimulus patterns which an instantiation of a neocognitron recognized correctly. The neocognitron was first trained with the patterns shown in the leftmost column.

as the input. This common output response within a class and distinct output response between classes is consistent even though different cells may be active in the S-plane and C-plane in the preceding layers (i.e., Input Layer, S_1, C_1, S_2, C_2). A collection of patterns which an instantiated neocognitron can recognize is presented in Figure 10.

The results presented in this section demonstrate that the research environment can accurately instantiate a neocognitron based upon parametric specification. This demonstration has been achieved by the reproduction of the results presented in (Fukushima and Miyake, 1982). To date, initial testing has been performed using the neocognitron in the character recognition domain. Subsequent testing is planned with additional network models and real-world images. One area from which images will be taken is the Automatic Target Recog-

nition (ATR) domain. The task of testing the research environment and neural networks instantiated for use in ATR applications is presently under investigation, and the preliminary results are are quite encouraging for future plans (Gilmore and Czuchry, Jr., 1990a; Gilmore and Czuchry, Jr., 1990b).

6 Future Directions

As mentioned previously in this paper, the environment has been designed to facilitate testing of neural network architectures on digitized images. The motivation for this effort has been an idea that by using real images as training patterns, network models such as the neocognitron should, theoretically, be able to extract 'useful' information. For example, through training the instantiated network by using a variety of images which contain a tree, the network should extract the common pattern of the tree and, thus, be able to indicate the presence of a tree in subsequent test images. A possible future research effort would investigate the size of various networks required to actually perform such recognition and to characterize any preprocessing requirements (e.g., use of Grossberg's Boundary Contour/Feature Contour System (Grossberg and Mingolla, 1985) as a preprocessor to simplify processing within the neocognitron itself). Additionally, the currently instantiated neocognitrons could be used with the incorporation of recent extensions to the neocognitron's architecture, i.e., feedback between layers (Fukushima, 1988), and could possibly provide for segmentation of trees within the test images after training had occurred. A significant amount of work would be required in order to obtain such results, but a real possibility of actually attaining them does exist.

Another consideration, given extensive use of the research environment, is the speed and versatility of processing related to the implementation of the environment. The present implementation of the model was developed on a Symbolics Lisp machine and was developed for in-house use. This lisp machine was chosen for its flexibility and power in symbolic manipulation and for its exploratory programming environment. The implementation language is Lisp. In order to enhance processing speed, there is a plan to port the environment to a SUN4-280. Additional speed enhancements may be achieved by converting list structures to arrays and/or reprogramming in the C language. Although the environment is currently organized to dynamically create an instantiation of Kunihiko Fukushima's neocognitron (Fukushima and Miyake, 1982), the versatility of the environment will be tested by instantiating additional neural network models.

7 Conclusion

This paper has discussed the successes and the corresponding challenges faced in the development of a generalized research environment for neural networks, particularly large scale networks such as the neocognitron (Fukushima, 1980). A variety of artificial intelligence and software modularization techniques have been shown to yield a research environment which can dynamically create (instantiate) neural network models. Although the results presented in this paper have concentrated on the research environment in the context of the neocognitron, the environment has been designed to incorporate the dynamic creation of a wide range of neural network models. The importance of the general-purpose nature is evident upon consideration of the task of research into new neural network models and architectures. With regard to neocognitrons in particular, the dynamically created model is based upon the

formulas and discussion presented in Fukushima's published works, especially (Fukushima and Miyake, 1982). The environment has been tested by instantiating this particular class of networks called neocognitrons. The dynamically created neocognitrons have been used to reproduce the general behavior of the varying excitation patterns within the hierarchy of Fukushima's pattern recognition architecture for computer vision (Fukushima and Miyake, 1982). The environment has also been used to used to test the neocognitron on real-world images (Gilmore and Czuchry, Jr., 1990a; Gilmore and Czuchry, Jr., 1990b). The main virtue of the neocognitron is that it is not sensitive to varying degrees of either of two fundamental pattern recognition limitations: 1. positional shifts, and 2. pattern deformation (e.g., due to noise or other forms of distortion). Future directions for investigating the neocognitron hold significant promise and include the addition of *feedback* into the instantiated neocognitron's hierarchy (Fukushima, 1988) as well as extensions to the neocognitron network to provide for handling 'real' images. An investigation is also being initiated to analyze the necessity of networks such as Grossberg and Mingolla's Boundary Contour/Feature Contour System (Grossberg and Mingolla, 1985) for preprocessing input data, when real images are used with the neocognitron. Future directions for the environment itself are based upon its use with a variety of network models. Additional neural network models will be instantiated within the environment and their behavior will be analyzed in a variety of pattern recognition tasks.

Acknowledgements

The overall research effort has been performed under the guidance of John Gilmore, Head of the Artificial Intelligence Branch at the Georgia Tech Research Institute. Instrumental assistance in the development of the initial routines used in the research environment has been provided by Harold Forbes and Stephen Strader. Dr. Ronald Arkin was fundamentally responsible for the idea of using the neocognitron as the first test case for the environment.

The author is indebted to Diane Czuchry, Dr. Kishore Ramachandran, and Dr. Pranas Zunde who have all made many helpful suggestions regarding this manuscript.

References

Alkon, D. L. (1984). Calcium-mediated reduction of ionic currents: A biophysical memory trace. *Science*, 226:1037–1045.

Alkon, D. L. (1989). Memory storage and neural systems. *Scientific American*, 261(1):42–50. July.

Alkon, D. L. and Rasmussen, H. (1988). A spatial-temporal model of cell activation. *Science*, 239:998–1005.

Carpenter, G. A. and Grossberg, S. (1987a). ART2: self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, 26:4919–4930.

Carpenter, G. A. and Grossberg, S. (1987b). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision Graphics Image Processing*, 37(1):54–115.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202.

Fukushima, K. (1988). A neural network for visual pattern recognition. *IEEE Computer*, 21(3):65–75.

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

Gilmore, J. F. and Czuchry, Jr., A. J. (1990a) An application of the neocognitron in target recognition. In *Proceedings of the International Neural Network Conference*, Paris, France.

Gilmore, J. F. and Czuchry, Jr., A. J. (1990b). Target detection in a neural network environment. In *Proceedings of SPIE Applications of Artificial Intelligence VIII*, Orlando, FL.

Grossberg, S. (1973). Contour enhancement, short term memory, and constancies in reverberating neural networks. *Studies in Applied Mathematics*, 52:213–257.

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, illusions. *Biological Cybernetics*, 23:187–202.

Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11:23–63. Reprinted in (Grossberg, 1988).

Grossberg, S., editor (1988). *Neural Networks and Natural Intelligence.* The MIT Press, Cambridge, MA.

Grossberg, S. and Mingolla, E. (1985). Neural dynamics of perceptual grouping: Textures, boundaries, and emergent segmentations. *Perception & Psychophysics*, 38:141–171.

Hebb, D. O. (1949). *The Organization of Behavior: A Neurophysiological Theory.* Wiley, New York.

Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Poc. Nat. Academy Sci, USA*, 79:2554–2558.

Lee, Y. C., Doolen, G., Chen, H. H., Sun, G. Z., Maxwell, T., Lee, H. Y., and Giles, C. L. (1986). Machine learning using a higher order correlation network. *Physica 22D*, pages 276–306.

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Spartan Books, Washington, D.C.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986a). Learning internal representations by error propagation. In *Parallel Distributed Processing*, volume 1, chapter 8, pages 318–362. The MIT Press, Cambridge, MA.

Rumelhart, D. E., McClelland, J. L., and the PDP Research Group (1986b). *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, volume 1: Foundations. The MIT Press, Cambridge, MA.

12-21-90 Technical Report

The Development of a Research Environment for Neural Networks: Instantiating Neocognitrons

Andrew J. Czuchry, Jr.

Georgia Institute of Technology
Artificial Intelligence Branch
Georgia Tech Research Institute
Atlanta, GA 30332

GTRI/AIB/TR: 901221

N/A

N/A

None

Unlimited

Neural networks can be thought of as combinations of generic pieces linked together in varying architectures. Many different models and architectures have been presented in the published literature. Networks may differ both in the characterization of their pieces and in the connection patterns of those pieces. In order to exploit the similarities between models, incorporate the differences between models, and automate the process of linking the pieces together, a prototype of a generalized research environment for neural networks is being developed. The main virtue of this generalized environment is the flexibility it provides for testing various neural network architectural and processing decisions without having to write programs. The viability of this research environment has been demonstrated by its use in the development of a generalized implementation of Kunihiko Fukushima's neocognitron. This paper initially introduces the generalized research environment, subsequently discusses the architecture of a test case network (the neocognitron), and finally presents the initial results in testing a neocognitron instantiated by the environment.

neural networks, instantiation, research environment, neocognitron, self-controlled processing

34 + title page

unclassified unclassified unclassified unlimited

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to **stay within the lines to meet optical scanning requirements.**

Block 1. Agency Use Only (Leave Blank)

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Names(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency. Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of ..., To be published in .... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availablity Statement. Denote public availability or limitation. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR)

DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE - See authorities
NASA - See Handbook NHB 2200.2.
NTIS - Leave blank.

Block 12b. Distribution Code.

DOD - DOD - Leave blank
DOE - DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports
NASA - NASA - Leave blank
NTIS - NTIS - Leave blank.

Block 13. Abstract. Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (NTIS only).

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.