A NETWORK OF NEURON-LIKE UNITS THAT LEARNS TO PERCEIVE BY GENERATION AS WELL AS REWEIGHTING OF ITS LINKS

Vasant Honavar and Leonard Uhr

Computer Sciences Technical Report #793

September 1988

To appear in: *The Proceedings of the 1988 Connectionist Models Summer School,* David Touretzky, Geoffrey Hinton, and Terrence Sejnowski (Ed.), Morgan Kaufmann, San Mateo, CA., 1988.

A NETWORK OF NEURON-LIKE UNITS THAT LEARNS TO PERCEIVE BY GENERATION AS WELL AS REWEIGHTING OF ITS LINKS

Vasant Honavar and Leonard Uhr

Computer Sciences Department
University of Wisconsin-Madison
Madison, WI 53706. U.S.A.

To appear in: *The Proceedings of the 1988 Connectionist Models Summer School*, David Touretzky, Geoffrey Hinton, and Terrence Sejnowski (Ed.), Morgan Kaufmann, San Mateo, CA., 1988.

A NETWORK OF NEURON-LIKE UNITS THAT LEARNS TO PERCEIVE BY GENERATION AS WELL AS REWEIGHTING OF ITS LINKS

Vasant Honavar and Leonard Uhr

Computer Sciences Department
University of Wisconsin-Madison
Madison, WI 53706. U.S.A.

Abstract

Learning in connectionist models typically involves the modification of weights associated with the links between neuron-like units; but the topology of the network does not change. This paper describes a new connectionist learning mechanism for *generation* in a network of neuron-like elements that enables the network to modify its own topology by growing links and recruiting units as needed (possibly from a pool of available units). A combination of generation and reweighting of links, and appropriate brain-like constraints on network topology, together with regulatory mechanisms and neuronal structures that monitor the network's performance that enable the network to decide when to generate, is shown capable of discovering, through feedback-aided learning, substantially more powerful, and potentially more practical, networks for perceptual recognition than those obtained through reweighting alone.

The *recognition cones* model of perception (Uhr1972, Honavar1987, Uhr1987) is used to demonstrate the feasibility of the approach. Results of simulations of carefully pre-designed recognition cones illustrate the usefulness of brain-like topological constraints such as near-neighbor connectivity and converging-diverging heterarchies for the perception of complex objects (such as houses) from digitized TV images. In addition, preliminary results indicate that brain-structured recognition cone networks can successfully learn to recognize simple patterns (such as letters of the alphabet, drawings of objects like cups and apples), using generation-discovery as well as reweighting, whereas systems that attempt to learn using reweighting alone cannot ever learn.

1. Introduction

There is currently a great deal of interest and effort in developing connectionist, neuronal, brain-like models for perception. Suitably connected networks with one or more layers of hidden units can, potentially, be trained using *error back-propagation* (Rumelhart1986) to produce mappings from sets of input patterns to the desired set of output patterns. Such systems assume a fixed network topology in which the weights associated with the links are changed as functions of the error between the output produced by the network and the output desired. While it is true that one layer of hidden units between the input and output units suffices, in theory, to enable the network to learn any possible desired input-output mappings, this may not be the most practical thing to do since the fan-in and the fan-out required of the units is arbitrarily large. It is desirable to restrict the connectivity of units to relatively small, preferably local, *receptive fields*. When receptive field sizes are limited in this fashion, multiple layers of hidden units become necessary to learn the desired input-output mappings. Few guidelines exist that suggest suitable numbers of units in each layer and good connectivity between the layers. It is therefore important that there be appropriate learning mechanisms with which the network can modify its own topology.

The ideal goal is a set of learning mechanisms that generate the simplest networks (according to some reasonable criteria of complexity) that are adequate for the given task. One approach to reducing the size of the target network is to incorporate an additional cost term into the criterion function minimized by the generalized delta rule that penalizes more complex networks (Rumelhart1988). In this paper, we take a different approach: The network starts with few links and a pool of available units; New links are grown and units are

This research was partially supported by the NSF and the University of Wisconsin-Madison.

recruited as needed. Standard error back-propagation is used to modify the weights associated with the links. Additional regulatory mechanisms maintain a balance between the addition of new links and the reweighting of existing links.

We begin by briefly describing recognition cones as an example of connectionist networks (Honavar1987) - networks of simple neuron-like units, structured into successively larger modules under topological constraints (e.g., near-neighbor connectivity, layered converging-diverging structure) suggested by the physics of the environment as well as the architecture of the brain (Uhr1972, Uhr1987). We present results from the simulation of carefully pre-designed recognition cone networks (with no learning) that perceive objects such as houses from digitized TV images. We outline different learning mechanisms that can be used in connectionist networks. We then show how *generation* that grows new links and recruits new units as needed, supplemented by standard error back-propagation, can develop networks satisfying certain topological constraints. We also present our first results of simulations that show the discovery of recognition cone networks that perceive simple patterns through feedback-aided learning - results that indicate how generation plus reweighting can improve performance substantially over reweighting alone.

2. Recognition Cones

This section briefly describes the overall architecture of recognition cones (Uhr1987, Honavar1987). The structure described here corresponds to one that is carefully designed (or programmed) for the task of recognition of specific objects or that which emerges as the result of learning - as explained later. (The emphasis is on visual perception, although some of the underlying principles appear to be of relevance to other sensory modalities.)

The architecture of recognition cones is suggested by the brain. The extremely complex human visual system is massively parallel, shallowly serial and roughly hierarchical, with functional organizations into larger structures of neurons interconnected by pathways that help to integrate diverse sources of information. Neurons usually (but by no means always) interact with near neighbors, and organize into successively larger structures (e.g., columns, hypercolumns, areas). Space does not permit a discussion of the human brain and visual system here; We simply refer the reader to the literature on these topics (Kuffler1984, Peters1986, Uhr1986, Crick1986, Sejnowski1986).

The basic building block of recognition cones is an adaptive neuron-like unit with a threshold or sigmoid output function which accepts inputs from other units via its input links, does some simple processing of the inputs, and sends out signals over its output links to the units into which it fires. Each unit in a given layer is connected to a small number of units in the adjacent layers. Large numbers of such units are organized, into a layered heterarchy of converging-diverging structures (hence the name recognition cones). It is converging because the spatial resolution of the layers decreases logarithmically as one moves up; It is diverging because a unit can link to several others in the layer above. The connectivity between layers is predominantly retinotopic (but gradually departing from retinotopy as we move up in the heterarchy).

Within each layer, each unit is linked primarily to nearby units in a relatively small surrounding neighborhood. This reflects the property of the real world that nearby points in the scene are likely to influence each other more than those that are farther apart. For simplicity, some Regularity could be imposed on the size and shape (and to make implementations on today's computers feasible it typically is - for example, each unit can link to its 4, 8 or 24 nearest neighbors in a square grid) of the neighborhoods. Or the connectivity patterns could model, albeit in a simplified manner, the structure of the retina, the lateral geniculate and the visual cortex (Uhr1986).

The input to the system is the image of the scene sensed by transducers (e.g., TV cameras) at the base layer of the recognition cones. The total system is made of several cone-like structures emerging from the retinal layer. There are several outputs from the system, typically, but not necessarily, from the higher levels (Figure 1 shows a schematic diagram of one such cone). Further, there may be a rich set of additional links, e.g., for feedback loops, between the different layers, including the output. (Computer simulations typically implement 2-way links, so that units can send out signals both upward and downward.)

It is probably worth pointing out here that recognition cones are closely related to *Pyramids* and other hierarchical architectures and algorithms that have been fairly extensively studied for image processing (Uhr1983, Uhr1986a, Burt1984, Rosenfeld1983, Dyer1987). Often the processors in a Pyramid are more powerful than the basic functional unit of the recognition cones. Recognition cones can be thought of as multi-apex multi-pyramids emerging from a common base, possibly augmented with additional links between the pyramids and decision networks at the top.

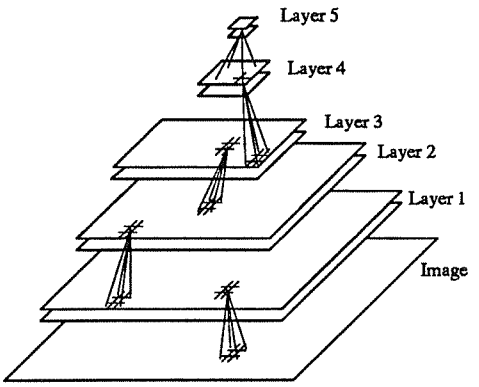

Figure 1: *Recognition cones: A Converging-diverging heterarchy of transforms (See text for explanation).*

Conceptually it is useful, for our purposes in this paper, to think of the adaptive neuron-like unit as an abstract process that computes one or more probabilistic or fuzzy transforms over its inputs. Such a unit typically has a small set of inputs which gather potentially relevant information, usually over a small compact window (figure 2) e.g., a region large enough to extract a local feature like an edge, angle, or (at a higher level where abstracted image arrays form the inputs to the arrays of processing units) contours, enclosures, and other higher-level features. Thus in figure 1 each cell in each layer has a number of such transforms. These are schematically shown as separate planes within each layer. A unit in a plane in layer L receives input from an n-tuple of windows picked from the planes in layer $L-1$.

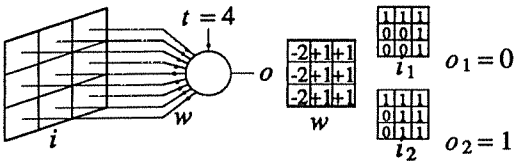

Figure 2: *A fuzzy transform over a* 3×3 *neighborhood: w is the mask of weights; t is the threshold;* i_1 *and* i_2 *are inputs in two* 3×3 *neighborhoods and* o_1 *and* o_2 *are the corresponding outputs.*

3. Designing Recognition Cones for Visual Perception

This section outlines how recognition cones are used for visual perception - the recognition of objects in the environment - within the constraints of the overall structures and processes described above. Results of simulation of the recognition cones model for perception of real-world objects are briefly presented. More detailed descriptions of the actual computer programs are found in (Uhr1979, Li1987).

Recognition cones are given a specific structure of transforms, as indicated by the following example (any cascaded structure of local processes can be used efficiently): The image is input into the retinal layer at the base of the pyramid. It is processed there with local smoothing (noise suppression) transforms and then by local gradient detectors - for example, a high-resolution difference-of-Gaussian operator. The next layer then looks for a family of edges at several different orientations, as well as color and textural features. The next layer combines oriented edges into corners, longer lines, curves, etc; colored regions into contrast-corrected larger regions; and so on. This process of successive transformation and merging of information to detect more and more complex features (figure 3) continues, possibly all the way to the top, until enough information is gathered so that specific objects are sufficiently highly implied by the features detected. In addition, continuing feedback from higher to lower layers activates processes at those layers (which may serve to gather additional evidence to confirm the implied features).

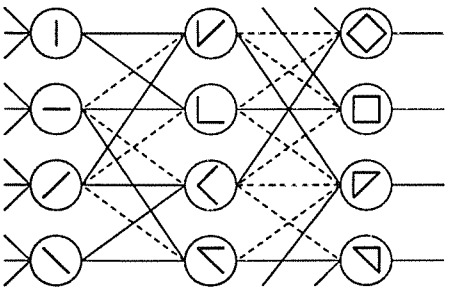

Figure 3: *Heterarchy of transforms detect increasingly complex features*

3.1. Performance of Pre-Designed Recognition Cones Used for Visual Perception

Recognition cone programs that apply sets of local fuzzy transforms, when given a small set of transforms (such as edges, angles and curves) recognize a variety of simple objects (squares, circles, etc.).

When given a large enough set of carefully chosen transforms distributed over 4-7 layers, such programs have demonstrated the capability to identify hand-printed letters (with gaps, small distortions and other forms of noise) as well as stylized hand-drawn sketches of place settings consisting of plates, spoons, knives and forks, and also some of the major structural features (e.g., windows) of photographed houses (Uhr1979).

Simulations of recognition cones which combine data-driven, bottom-up processing where many feature-detecting transform are applied in parallel with model-driven, top-down processes which are activated when certain transforms respond to the image with sufficiently high weights (Li1987) recognize complex real-world objects such as windows, shutters, doors, houses, etc. from digitized (grey values range from 0 to 255), high resolution (512×512) TV images of outdoor scenes. The program was tested on three scenes, each containing a different house (two of the scenes were used by the programmer in determining the set of transforms to be provided to the program and the third was used to evaluate the generality of the transforms) and a fourth scene containing an office building (Figure 4 shows these scenes) with good results in identifying the building and its major structural components. Figure 5 shows some of the results (Li1987).

Thus, recognition cones, although they are highly parallel, and also neuronal and largely connectionist - albeit with additional more global brain-like structures, have been shown able to handle complex vision problems at least as well as do computer vision systems that rely on explicit serial model-matching, and are, as a consequence, much slower and, in most cases, rather brittle and difficult to extend to a full-blown vision system, which must handle the much larger number of object-classes, each with a much larger number of possible variant object-instances. Their power and extensibility, along with their micro-modular neuron-like architecture, make recognition cones an interesting test-bed for the study of perceptual learning.

4. Learning

Learning refers to the acquisition of new knowledge; the development of perceptual, motor, and cognitive skills through instruction or experience; the organization and integration of acquired knowledge into effective representations; and the discovery of new facts, theories, or ideas through observation, experimentation, and thought (Michalski1983, Uhr1973).

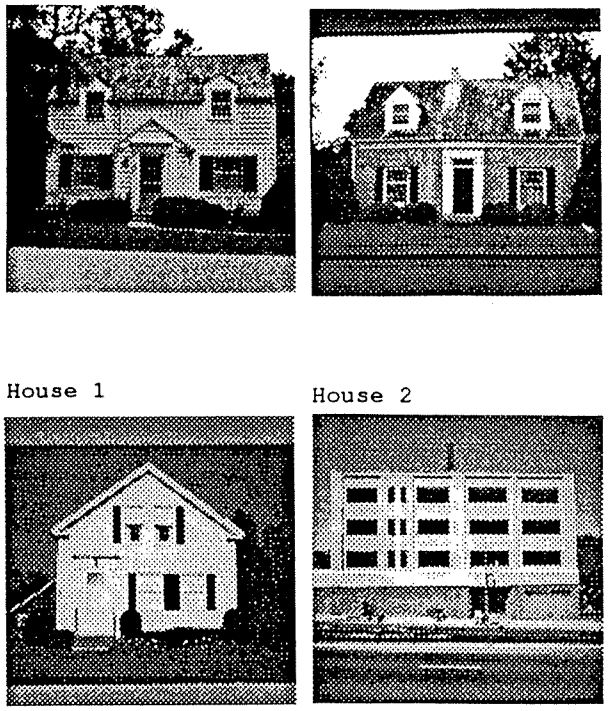

Figure 4: *Digitized 512×512 pictures of buildings that were recognized by the multi-layered image and model-driven recognition cones program.*

Possible window areas					
-	W1-6	W10	W11-12	N4	N5
B (elong)	0.50	0.50	0.50	0.30	0.50
B (text)	0.40	0.40	0.40	0.40	0.00
B (left-bd)	0.60	0.60	0.60	0.00	0.10
B (right-bd)	0.60	0.10	0.60	0.60	0.30
B (window)	0.45	0.38	0.45	0.34	0.20
B (v-sibling)	0.60	0.60	0.60	0.00	0.00
B (h-sibling)	0.60	0.60	0.60	0.00	0.60
B' (window)	0.49	0.46	0.49	0.13	0.20

Figure 5: *Results of identifying windows in the office building. W1 through W12 correspond to the 12 windows in the office building (W7 through W9 are not shown in the table); N4 and N5 correspond to 2 of the several regions in the scene that do not contain a window; B (window) and B' (window) correspond respectively, to the evidence for a window before and after relaxation triggered by the model-driven, top-down processes.*

4.1. Learning as Constrained Induction

Learning entails the building of usable models of the environment in which the learner-perceiver (whether natural or artificial) operates. Given a sufficiently rich environment, one that captures at least a significant portion of the great complexity and variety present in the real world, the number of possible inputs and the number of possible structures relating and combining them is enormous (If there are N inputs, each capable of taking V values, the number of possible structures is V^N). Only a small fraction of these associations is meaningful in modeling the environment. This suggests that the perceptual learning system should be designed so that it is either equipped with - or develops through learning - structures that enable it to detect and respond to the features, and the relationships among features, in the environment needed to handle the tasks it has to perform.

Given a certain structure, or a set of structural constraints for the development of the perceptual system, knowledge of the environment is gained by a process of induction (constrained by the structure of the system) applied to the information provided by the senses. Induction is the process by which a system develops an understanding of principles or theories that are useful in dealing with the environment by generalization and specialization from specific examples or instances presented to it (Michalski1983, Holland1986). This includes the process of experimentation and discovery, that is, the setting up of hypotheses and then the accumulating of evidence to confirm or deny their validity.

4.2. Basic Neuronal Mechanisms for Learning

Learning in a neuronal system may involve modification of any of the following:

[1] The processing functions of the nodes (e.g., change in the threshold or output function),

[2] The weights (or transfer functions) of the links,

[3] The topology of the network, and

[4] The learning rules themselves

All are being investigated; however, this paper is restricted to examining [2] and [3].

Learning by changing the weights associated with the links can be accomplished in a neuronal network in a number of ways including a Hebbian rule (Hebb1949), the generalized delta rule (Rumelhart1986), etc. Changing the topology of the network involves the growth of new links, the recruitment of new units (possibly from a pool of available units), or the deletion of existing links. Any of these forms of learning may or may not use feedback from a *teacher* or the environment. This paper deals with networks that learn to classify patterns when feedback tells the network what the right response should have been for a given pattern.

4.3. The Learning of Useful Transforms through Generation and Reweighting

As noted earlier, the adaptive neuron-like unit can be viewed as an abstract process - a transform, that computes one or more probabilistic or fuzzy functions over its inputs. In such a system, learning by induction can be viewed as the generation, tuning, and retention of a set of transforms that are adequate for the perceptual tasks demanded of the system. The system learns, or is initialized with (as though by evolution) a set of low-level transforms such as edge detectors, color detectors, etc. What follows is a general description of the mechanisms; A particular implementation is explained in detail in the next section. Transforms are modified by changing the weights of their implieds (that is, the output links of the adaptive neuron-like units) and their conditionals (the input links of the units) according to one of the standard reweighting rules; and by changing thresholds of firing or the output functions of the units. New transforms are added by *generation* - which involves the growth of new links between units (implieds, conditionals, with appropriate weights, which are themselves learned), or/and new units (possibly recruiting them from a pool of available units). Thus the network learns - through both generation and reweighting - a set of fuzzy transforms adequate to classify the training patterns correctly, to the desired degree of accuracy - and, because of their probabilistic structures, the much larger set of possible instances the program must handle, and on which it is tested.

4.3.1. The Need for the Capability to Generate New Transforms

The input to the network represents a certain encoding of the environment. A single layer of neuron-like units computing fuzzy transforms over this encoding is combinatorially explosive and not always sufficient to produce the desired input-output mapping (Minsky1969). Internal representations that capture non-linear relationships between features in the input encoding must be created to overcome this problem. While it is true that one layer of hidden units between

the input and output layers theoretically suffices to enable the network to learn the desired input-output mappings, the fan-in and fan-out required of the units is arbitrarily large. Large fan-in and fan-out imply longer links, a higher density of links, and hence a much greater, combinatorially explosive, cost/complexity. In the worst case, NV^N links are needed for N pixel input images, each pixel taking one of V possible values, a quite impossible number even for toy images (e.g., 8×8), much less the 256×256 to 4092×4092 arrays needed for real-world images. To fend off this combinatorial explosion it is essential to restrict the links to relatively small *receptive fields*. When receptive field sizes are limited, multiple layers of hidden units become necessary to compute global functions and to represent the non-linear relationships between features in the input encoding. It is difficult, and in practice impossible except for trivial cases, to foresee the necessary connectivity, and the number and the depth of transformations (which corresponds to the number of layers in the network if no cycling between layers is permitted) needed for a particular task on which the network is to be trained; this is especially true when, in dynamic real-world environments, the task changes over time and learning can never cease.

Only if the network has adequate initial connectivity, can reweighting guided by feedback eventually find the right set of weights that would result in correct classification of patterns in the training set. Thus the generation of new transforms is an essential learning mechanism in networks where the necessary connectivity cannot be established in advance, because no amount of reweighting would enable such a network to correctly classify all the patterns in its training set.

4.3.2. Modification of Existing Transforms

Feedback is used to weaken the links of the transforms that implied the *wrong* thing (and, optionally, to strengthen those that implied the *right* thing), by propagating the information from the unit or units that made the choice, usually moving from the output layer of the network, backward through the network, down to the input layer. At the output layer, a node that made the wrong choice releases a *transform down-weight signal* that weakens the links that fired into it from the nodes at the next lower layer. This down-weight signal is propagated back through the network until the input layer is reached. Every node that receives a down-weight signal, weakens the links that fired into it on the training presentation. In a similar fashion, a node at the output layer that would have been correct (had it fired) releases a *transform up-weight signal* that strengthens the links that fired into it from the next lower layer and this up-weight signal is propagated back through the network. This is the form of learning that has been studied widely in connectionist systems, and several algorithms are available for this purpose (Hinton1987). The one we use is very similar to the *generalized delta rule* (Rumelhart1986), also known as the *error back-propagation* algorithm.

4.3.3. The Generation of New Transforms

The generation of a new transform is triggered by negative feedback under certain appropriate conditions (which will be explained later). Suppose the feedback indicates that the system implied the *wrong* thing. This is triggers the release of a *transform generation signal* by units that received negative feedback, which is transmitted to units successively in the layers below that contributed inputs to the units in question - just as the error signal is propagated back for reweighting. At one or more layers, a subset of the units receiving the transform generation signal recruit one or more unused units from the next-higher layer by growing a link to that unit. Growth of these links takes place without violating the topological constraints (such as those of predominantly near neighbor connectivity, retinotopy, convergence, and layered organization) imposed by recognition cones. The effect of this is to add a new transform to the existing set. The transforms so added participate in the learning process according to the same principles as those described above. The conditions under which new transforms are added through generation, instead of simply reweighting the existing transforms, will be explained later.

4.3.4. The Discarding of Poor or Useless Transforms

Transforms get discarded either by a gradual lowering of weights as a consequence of negative feedback (when the weight on the link reaches a value close to zero, the link is broken) or by an abrupt breaking of some of the links, under the influence of appropriate regulatory mechanisms. The discarding of transforms that are deemed poor or useless creates space in the system, by freeing up units that may then be used in the generation of new (and hopefully better) transforms to replace them. The conditions under which it is appropriate for the network to discard transforms are discussed later.

4.4. Regulatory Mechanisms that Decide when to Generate and when to Discard Transforms

A network that both reweights and generates must somehow strike a reasonable balance between these two learning mechanisms. If learning is restricted to reweighting alone, the network may never be able to achieve the desired performance level of recognition, because of reasons outlined earlier. On the other hand, if new transforms are generated each time negative feedback is received by the network the essential process of tuning of transforms by reweighting is disturbed and the network is likely to end up with a large set of transforms most of which are only rarely useful. Similarly, discarding existing transforms has to be done when appropriate. Regulatory mechanisms that decide when to generate new transforms and when to discard existing ones are therefore needed.

One possible mechanism to decide when to generate a new transform is suggested by the need to guide (and possibly goad) the network in the direction of developing into the simplest possible structure (according to some criteria of complexity), that does not violate the topological constraints placed on the network (such as, e.g., the size of a receptive field) that is adequate to perform the pattern classification tasks for which the system is being trained. We call this the *minimal complexity heuristic*. Several measures of network complexity suggest themselves, e.g., the number of transforms (number of nodes, links, or both), the depth of transformations from input to output (number of layers, which in turn determines the time for processing a given input pattern).

One such regulatory mechanism, based on the *minimal complexity heuristic* that decides when to add new transforms has been implemented in the simulation to be described below. This uses the number of transforms in the network as a measure of complexity. Thus, we seek networks with the smallest number of transforms adequate for classifying the patterns in the training set correctly. This suggests that the network should continue to reweight existing transforms so long as its performance is improving; and generate a new transform when performance ceases to improve with reweighting alone (and the network has not yet reached the target performance). This requires the network to have some mechanism to keep a (recent) portion of the learning curve for each pattern class on which the network is being trained. The details of implementation are described later. Other regulatory mechanisms based on different measures of network complexity are possible, and are being investigated.

Regulatory mechanisms are needed to decide when to discard a transform. Discarding a transform by breaking its input and output links may appear to be a drastic step, but it is necessary if the network's performance remains consistently poor over intolerably long periods of time (and reweighting and prior generation have failed to give the desired improvement in performance), or when it becomes difficult to grow new links needed to generate new transforms, without violating the topological constraints on the network (because most of the allowed units and links have been used up).

Additional regulatory mechanisms may be needed to determine where in the network to generate new transforms (e.g., in which layer); or to decide which transforms to discard (a transform with fewer output links leaves the network relatively undisturbed). These issues are interesting in their own right and deserve further examination.

The realization of regulatory mechanisms that guide the generation, modification, and retention of transforms requires structures that maintain, update and transmit information (accumulated through local computations) concerning the performance of the network at the task that it is being trained for. Such information may include some measure of the history of the network performance which is used to determine the nature and the extent of changes to be made with learning. If the system has been responding correctly most of the time in the recent past, it perhaps should be conservative in adding, deleting and modifying transforms as a result of feedback. On the other hand, if the system has been responding incorrectly most of the time in the recent past, it should perhaps be more radical in making those changes.

The mechanisms of reinforcement of *good* transforms, and the generation of new transforms explained above, over a period of time result in the development and retention of sets of fuzzy transforms that are useful for recognizing the objects in the environment. On the other hand, transforms found not useful will fade away, since they are negatively reinforced, (or, as is occasionally the case, discarded) by the learning mechanism. Several questions remain to be answered. For example: What is an appropriate set of regulatory mechanisms? Can higher order control be exercised by the network, as a function of its performance, on the nature of the particular regulatory mechanisms that come into play?

5. Simulation of Discovery of Recognition Cone Networks by Generation and Reweighting

This section describes and gives first results of a computer simulation that uses generation and reweighting of transforms, as a function of feedback, supplemented by mechanisms that aid the network in deciding when to generate, in its search to discover successful recognition cone networks.

5.1. Topological Constraints and Network Structure

The retinal layer can be set to any size for a particular run. In most of the runs made to date, this resolution has been 32×32 and the first layer has been provided with a set of eight oriented edge detectors, spaced 45^{o} apart. Pre-programmed transforms are not needed, since the same learning algorithm can be applied to this first layer so that these edge-detectors, and possibly others that are useful, are learned as well. All the higher layers of the network are initially empty; Units recruited by the network as it generates new transforms as a result of learning get embedded in these layers. This generation and embedding of transforms occurs without violating the topological constraints, such as near neighbor connectivity, on the structure of the network.

Transforms (the exact number at any given time is a function of the system's past experience) are applied at each location in a given layer. The outputs of the transforms are mapped retinotopically into the next layer. The resolution of layers goes down by a factor of 2 (giving 2×2 logarithmic convergence) moving up from the retinal layer. A transform at any layer computes a function over the outputs of one or more transforms within a 2×2 window (except at the first layer, where it is 3×3) at the layer below. For the sake of simplicity, the same set of transforms is uniformly applied at all locations in each layer. This also means that when a new transform is learned at some location in a layer, it is duplicated for use at all locations in that layer. This seems logically compelling for a general-purpose recognizer, since a transform that is useful at some particular location is likely to be equally useful at all locations. Space does not permit a discussion of possible neuronal mechanisms that might duplicate the transforms learned at one location in a given layer at other locations in that layer. It is worth pointing out however, that such mechanisms may be implemented by broadcasting the transform, either laterally through links in its own layer or by passing it up through the pyramid-like converging structure to the top of the network and then broadcasting it back down to all the locations in the layer in which the transform was generated.

5.2. The Learning Algorithm and its Implementation

The network is trained by inputting images containing object-instances to its retina, associated with feedback. The system's two basic learning mechanisms are the reweighting of existing transforms as a function of feedback and the substantially less frequent discovery of non-linear relations between features at a layer and the resulting generation of a higher level transform that detects such interactions.

Reweighting of existing transforms as a result of feedback follows a form of the *generalized delta rule* (Rumelhart1986). Suppose the pattern class C_W is implied by the network with a weight W_W and the pattern class indicated by the feedback, C_R is implied by the network with a weight W_R, (that is, $W_W > W_R$), the total amount of reweighting is limited to $(K \times (W_W - W_R))$ where K is a parameter that is related to the *learning rate*. Our current implementation has K set equal to 0.25.

Subsequent paragraphs describe the simple version of the *minimal complexity heuristic* mentioned earlier, that the network uses to decide when to generate new transforms. The current implementation does not have any mechanisms for discarding *poor or useless* transforms (except that the gradual reduction in their weights as a result of feedback means they will have little or no effect on the recognition task).

A new transform incorporating non-linear interactions between features is added by the learning algorithm based on a global evaluation of the network's performance with respect to a specific pattern class. Such an evaluation function is straightforwardly computed using neuron-like elements. It is simply the ratio of correct responses to the total number of training instances of a certain pattern class averaged over a number of presentations which we call the *interval*, I (I is currently set to 10). This value is updated on every presentation of a training example. Thus there is an evaluation of the network's performance with respect to each pattern class known to the network. The values of this evaluation function for two consecutive intervals (previous and current) E_P and E_C are maintained, and updated as appropriate, by the network.

A small randomly chosen, structurally near (because of near-neighbor connectivity constraint on the topology) subset of non-linear interactions between features is added to the set of transforms at some of the layers (currently, only one transform is added at the first

layer from the top where such a generation can take place) when the pattern class indicated by the feedback - C_R is different from the one output by the network if:

[1] The improvement in the network's performance with respect to the pattern class C_R over a period of time as indicated by the difference in the values of the evaluation function for the pattern class C_R for the current and previous intervals is lower than a predetermined minimum, and

[2] The performance has not yet reached the desired level - as indicated by the value of the evaluation function for the current *interval* for the pattern class C_R.

For example, if transforms T_1 and T_2 respond to a particular instance of pattern class C_i, and if the network responds incorrectly to that instance (that is, the network's response is C_j where $j \neq i$), and if the two conditions mentioned above are satisfied for C_i, a new transform that responds when both T_1 and T_2 respond is generated at the appropriate layer. Thus these transforms capture important relative spatial relations - such as for example, between a horizontal edge and a vertical edge necessary to constitute a particular kind of corner angle. There may be several candidates for such an addition, in which case one of them is chosen at random. This mechanism is easily generalized to handle non-linear interactions across n-tuples of features (instead of just pairs of them) at any location in a given layer.

When a pattern belonging to a previously unseen object-class is presented to the network during training, an output node is recruited at the decision layer, and a set of links grown from that node to the outputs of transforms that responded to the example of the new pattern class. >From then on, learning proceeds as described earlier.

Space does not permit a detailed discussion of connectionist network designs that implement all aspects of this simulation (e.g., computing and maintaining the evaluation of network performance over time). The functional description of such networks (e.g., using E_P and E_C to obtain the slope of the learning curve for a pattern class) must not be interpreted too literally. They are best treated as first approximation functional modules, details of which will be worked out as through simulation and analysis.

5.3. Results of the Simulation

Networks to classify simple patterns (such as 2-dimensional line drawings of simple objects like cups, apples, bananas (in a 24×24 sub-array of the 32×32 retina) have been constructed by the learning program with a few (4-12) *epochs* of training where an *epoch* involves presentation of a few instances (3-5) from each pattern class. Figure 6 shows a sample set of object instances used. The networks so constructed by the learning program were tested with 3-4 previously unseen patterns belonging to the pattern classes used during learning. Test results indicate 100 percent accuracy of recognition on these patterns. Relatively small numbers (10-12) of transforms discovered by the learning program at higher layers (layers 2 through 5) were adequate for this purpose.

The training and test runs were conducted by randomly dividing the set of instances provided by three volunteers into two subsets, one for training and the other for testing. The drawings were made using the *xgremlin* graphics utility on a Digital VAXstation 3200. Figure 7 shows several instances of two of the object classes; Binary versions of such drawings on a 32×32 grid were used in the runs.

100 percent accuracy of recognition was also obtained with runs made using three letters of the alphabet (T, D, E). Additional runs were made with six pattern classes obtained by combining the two sets mentioned above, with similar results. Some of the results obtained from these runs (both on training as well as test sets) are summarized in figure 8. These results indicate the feasibility of developing recognition cone networks through learning by generation-discovery.

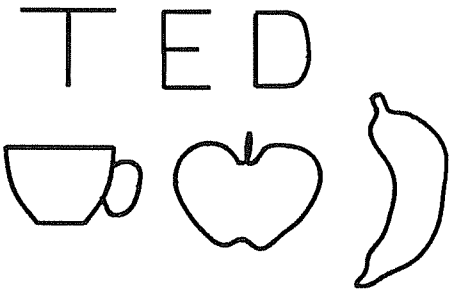

Figure 6: *Sample images used in the simulation of learning*

When runs were repeated with learning restricted to only reweighting (i.e., no generation was allowed) of the eight oriented edge detectors (corresponding to a single layer of hidden units, with local receptive fields) the system's performance leveled off at roughly 50-75%

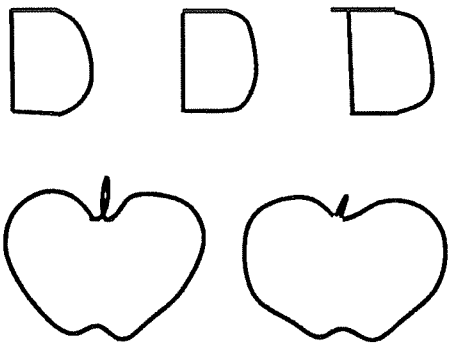

Figure 7: *Sample instances of two of the object classes used in the training and test sets.*

Summary of Runs				
Run	Classes	# of training instances / class	# of test instances / class	Generation
1	T, D, E	4	3	Off
2	T, D, E	4	3	On
3	Cup, Apple, Banana	4	4	On
4	T, D, E, Cup, Apple, Banana	4	3	On

Summary of Results				
Run	# of training epochs	# of transforms generated	% accuracy on training set	% accuracy on test set
1	10	0	75	67
2	6	6	100	100
3	7	8	100	100
4	12	12	100	100

Figure 8: *Summary of runs (1 through 4) and the corresponding results of simulation of learning with generation and reweighting.*

success, rather than achieving 100% success. Figure 9 shows a comparison of the plots of recognition accuracy against the number of training epochs with and without generation on both training and test patterns in the (T, D, E) case. This confirms the fact that non-linear spatial interactions between lower level features often must be detected, but cannot be, in less-than-completely-connected networks. Generation makes it possible for the network to discover such non-linear interactions where necessary.

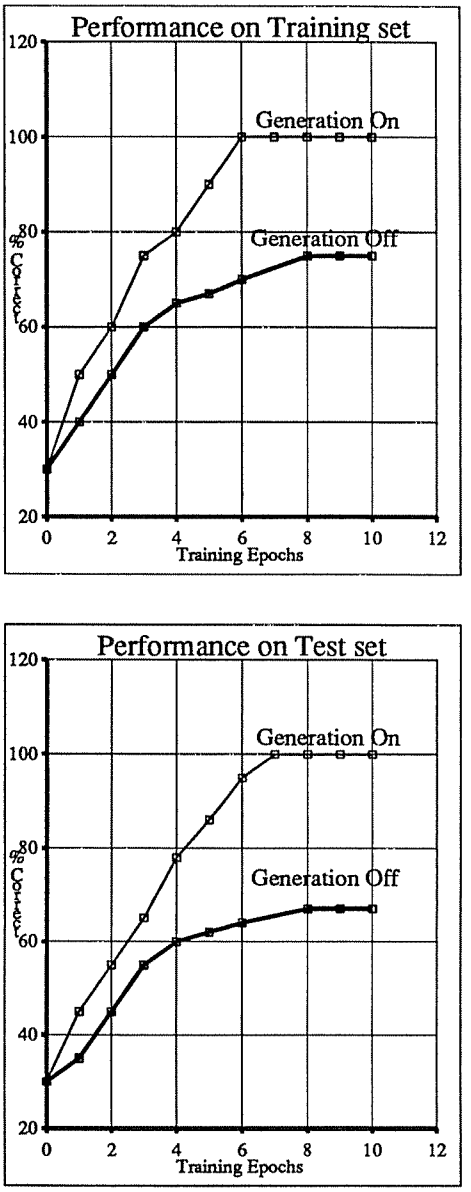

Figure 9: *Performance on training and test patterns* (T, D, E) *with and without generation (runs 1 and 2) in recognition cones.*

More extensive evaluation of the learning program on larger sets of more complex patterns, as well as comparisons with other systems (with and without local receptive fields or convergence, with varying amounts of connectivity, with and without generation), is in pro-

gress. Space does not permit a discussion of these runs here. A preliminary examination of the results seems to suggest that networks with local receptive fields, converging topology and the capability for generation yield significantly better learning than those without one or more of these features (Honavar1988).

6. Discussion

This section discusses briefly, the impact of the topological constraints, generation, and the regulatory structures on learning in connectionist networks in the context of the results presented in this paper.

6.1. Generating, Reweighting, and Discarding Transforms

A combination of generation - a mechanism involving growing or recruiting new links and nodes - plus reweighting by error back-propagation appears to yield more powerful learning than reweighting alone in multi-layered connectionist networks. Feedback-triggered generation and reweighting, supplemented by regulatory mechanisms that decide when to generate, operating under brain-like topological constraints on the network, produce network structures that successfully recognize simple pattern classes, as indicated by the preliminary simulation results presented in this paper. The regulatory mechanisms are motivated by the desirability of generating the simplest possible networks capable of accurate recognition. A more complete system should probably have mechanisms for discarding useless transforms to create space for potentially better transforms. This would have to be supplemented by regulatory mechanisms that decide which transforms to discard, and when. Generation and discarding of transforms enable the network to modify its own topology as a function of learning, in response to (possibly changing) environments.

6.2. The Impact of the Minimal Complexity Heuristic on Generalization - A Conjecture

It is tempting to speculate as to how generation might impact on the generalization properties of the network. Better generalization is considered a desirable characteristic of connectionist networks. Related work (Hinton1987a) seems to suggest that generalization in connectionist networks is sensitive to the number of hidden units. If there are many more hidden units than needed, the network might generalize rather poorly; if there are too few, the network may never reach the target performance desired. The minimal complexity heuristic, that decides when to generate, forces the network to reweight and tune the transforms so long as performance continues to improve, before generating a new transform (and, possibly, recruiting a new hidden unit). Generation and discarding of transforms may be viewed in this context as providing a dynamic mechanism for regulating the number of hidden units, leading to the conjecture that networks that generate, discard, and reweight transforms may exhibit good generalization properties as well. Generation makes possible the linking up of enough units to handle a problem; minimal generation tends toward the smallest necessary number (given the constraints on the topology), hence better generalization.

6.3. The Impact of the Constraints on the Network Topology on Learning

The constraints on the network topology determine the space of possible transforms that can be learned. It also biases the network so that the learning of certain relations is favored. It is instructive to examine how the topological constraints in recognition cones (retinotopic mapping, near-neighbor connectivity, converging-diverging heterarchy) interact with the learning mechanisms.

Retinotopic mapping and near neighbor connectivity exploit spatio-temporal contiguity in the environment. This favors the discovery and learning of relations between subpatterns that are imaged onto neighboring regions of the retina. For example, if the object imaged is a chair, its sub-parts (e.g., legs, seat, backrest) are projected onto the neighboring parts of the visual field with all the spatial relations between them intact.

Since each layer fires into the next, the learning of structure is a hierarchical, repeated operation (in the layers as well as over several presentations of the patterns). Generation ensures that successively more complex non-linear relations between features in the input encoding of patterns are discovered at higher layers, to be assessed by the new transforms that are added. For example, the lower layers might learn the associations between several vertical edges more or less aligned with each other and thus discover (learn) the concept of a long vertical line. At higher levels, a vertical line and a horizontal line that intersect facilitate the learning of the more complex concept of a corner, and so on. The effects of this are two-fold: The learning of simpler concepts precedes the learning of more complex concepts and successively more global relations are learned at

successively higher layers. This is confirmed by the transforms generated by the network simulation, since they exhibit these characteristics.

6.4. Self-monitoring and Self-Evaluation to Guide Learning

Structures that maintain, update, and as appropriate transmit information about the network's performance over a sequence of training presentations (e.g., a portion of the learning curve for each pattern class) enable the network to constantly monitor and evaluate changes in its performance. Such an evaluation helps the network to decide on particular learning strategies (e.g., generation versus reweighting) - as indicated by the results of our simulation. It may also be used to dynamically alter parameters such as the *learning rate* as appropriate in connectionist networks.

7. Conclusions

Connectionist networks built from simple neuron-like units arranged in brain-like topologies can be constructed to yield relatively good perceptual recognition of complex real-world objects in large images.

The preliminary results presented in this paper suggest the possibility of discovering such networks, by realizing significantly more powerful and potentially more practical learning than that given by reweighting alone, through a combination of:

[1] Different learning mechanisms - generation, reweighting, (and when necessary discarding) of transforms,

[2] Regulatory mechanisms - using minimal complexity heuristics, and

[3] Brain-like constraints on the network topology - near-neighbor connectivity, converging-diverging heterarchy.

References

Burt1984.
Burt, P. J., "The pyramid as a structure for efficient computation," in *Multiresolution Image Processing and Analysis*, ed. A. Rosenfeld, Springer-Verlag, Berlin, 1984.

Crick1986.
Crick, F. H. C. and Asanuma, C., "Certain aspects of the anatomy and physiology of the cerebral cortex," in *Parallel Distributed Processing, vol. 2: Psychological and Biological Models* , The MIT Press, Cambridge, Massachusetts, 1986.

Dyer1987.
Dyer, C. R., "Multiscale image understanding," in *Parallel Computer Vision*, ed. L. Uhr, Academic Press, New York, 1987.

Hebb1949.
Hebb, D. O., *The Organization of Behavior*, Wiley, New York, 1949.

Hinton1987.
Hinton, G. E., "Connectionist learning procedures," *Technical Report CMU-CS-87-115*, Computer Science Department, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1987.

Hinton1987a.
Hinton, G. E., "Learning translation invariant recognition in a massively parallel network," in *PARLE: Parallel Architectures and Languages, Europe. Lecture Notes in Computer Science*, ed. G. Goos, J. Hartmanis, Springer-Verlag, Berlin, 1987.

Holland1986.
Holland, J. H., Holyoak, K. J., Nisbett, R. E., and Thagard, P. R., *Induction: Processes of Inference, Learning, and Discovery*, The MIT Press, Cambridge, Massachusetts, 1986.

Honavar1987.
Honavar, V. and Uhr, L., "Recognition Cones: A neuronal architecture for perception and learning," *Computer Sciences Technical Report #717*, Computer Sciences Department, University of Wisconsin-Madison, Madison, Wisconsin, September 1987.

Honavar1988.
Honavar, V. and Uhr, L., *Generation-discovery in brain structured networks (in preparation)*, 1988.

Kuffler1984.
Kuffler, S. W., Nicholls, J. G., and Martin, A. R., *From Neuron To Brain*, Sinauer Associates Inc., Sunderland, Massachusetts, 1984.

Li1987.
Li, Z. N. and Uhr, L., "Pyramid vision using key features to integrate image-driven bottom-up and model-driven top-down processes," *Systems, Man and Cybernetics*, vol. 17, pp. 250-263 , March 1987.

Michalski1983.
Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. (eds.), *Machine Learning - An Artificial Intelligence Approach, vol. 1*, Tioga , Palo Alto,

California, 1983.

Minsky1969.
Minsky, M. and Papert, S., *Perceptrons: An Introduction to Computational Geometry,* The MIT Press, Cambridge, Massachusetts, 1969.

Peters1986.
Peters, A. and Jones, E. G. (eds.), *Cerebral Cortex: Vol. 3. Visual Cortex,* Plenum, New York, 1986.

Rosenfeld1983.
Rosenfeld, A., "Pyramids: multiresolution image analysis," *Proceedings of the Third Scandinavian Conference on Image Analysis*, pp. 23-28, July 1983.

Rumelhart1986.
Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning internal representations by error propagation," in *Parallel Distributed Processing vol. 1: Foundations*, The MIT Press, Cambridge, Massachusetts, 1986.

Rumelhart1988.
Rumelhart, D. E. (Work in progress: Connectionist Models Summer School Lecture, 1988), 1988.

Sejnowski1986.
Sejnowski, T. J., "Open questions about computation in cerebral cortex," in *Parallel Distributed Processing, vol. 2: Psychological and Biological Models*, The MIT Press, Cambridge, Massachusetts, 1986.

Uhr1972.
Uhr, L., "Layered recognition cone networks that preprocess, classify, and describe," *IEEE Transactions on Computers*, vol. 21, pp. 758-768, 1972.

Uhr1973.
Uhr, L., *Pattern Recognition, Learning and Thought,* Prentice-Hall, Englewood Cliffs, New Jersey, 1973.

Uhr1979.
Uhr, L. and Douglass, R., "A parallel-serial recognition cone system for perception," *Pattern Recognition*, vol. 11, pp. 29-40, 1979.

Uhr1983.
Uhr, L., "Pyramid multi-computer structures, and augmented pyramids," in *Computing Structures for Image Processing*, ed. M. J. B. Duff, Academic Press, London, 1983.

Uhr1986a.
Uhr, L., "Multiple image and multi-modal augmented pyramid networks," in *Intermediate Level Image Processing*, ed. M. J. B. Duff, Academic Press, London, 1986.

Uhr1986.
Uhr, L., "Toward a computational information processing model of object perception," *Computer Sciences Technical Report #651*, Computer Sciences Department, University of Wisconsin-Madison, Madison, Wisconsin, July 1986.

Uhr1987.
Uhr, L., "Highly parallel, hierarchical, recognition cone perceptual structures," in *Parallel Computer Vision*, ed. L. Uhr, Academic Press, New York, 1987.