A Connectivity Analysis of a Class of Simple Associative Neural Networks

Dan Hammerstrom[1]
Oregon Graduate Center
19600 N.W. von Neumann Dr.
Beaverton, Oregon 97006

Technical Report No. CS/E-86-009
January 1988

Abstract

The efficient realization using current technology of Very Large Connection Networks (VLCN) with more than a billion connections requires that these networks have a high degree of communication locality. A network exhibits *locality of communication,* if most of its processing elements connect to other physically adjacent processing elements in any reasonable mapping of the elements onto a planar surface. Real neural networks exhibit significant locality, yet most connectionist/neural network models have little. In this report, a network model is developed based on communication theory. This model is then used to analyze the connectivity requirements of simple association. Several techniques based on communication theory are presented that improve the robustness of the network in the face of sparse, local interconnect structures. Also discussed are some potential problems when information is distributed too widely.

[1] This work was supported in part by the Semiconductor Research Corporation contract no. 86-10-097, and jointly by the Office of Naval Research and Air Force Office of Scientific Research contract no. N00014 87 K 0259.

1. INTRODUCTION

Neural networks models have been proposed as a new paradigm of cognitive processing [FeB82,Ham86,HMT86,RuM86]. These networks exhibit massive parallelism and fault-tolerance and for these reasons offer the possibility of significant increases in cost/performance for systems that do speech and vision analysis at the person-machine interface. They operate by generating multiple, parallel hypotheses of their input and then relaxing to a single, best-match hypothesis (interpretation of input). This mode of operation allows biological neural networks to exhibit cognitive processing times of hundreds of milliseconds from unreliable components (neurons) whose switching times are in the tens of milliseconds.

An early example of this work was the Perceptron. It was shown that a single level (row) of Perceptrons was limited in the type of pattern recognition tasks it could perform [MiP69] (which is not surprising, since one cannot compute much with one level of computer logic either), but efficient multi-level learning algorithms for perceptron-like networks were not obvious at the time.

Recently, a variety of techniques have been proposed by several researchers to do multi-level learning. The most prominent have been Barto's statistical learning [Bar85], the Boltzmann machine [HSA84], Fukushima's multi-level pattern recognition networks [FMI83], Grossberg's adaptive networks [CaG86,CoG83,Gro80,Gro82], and the programming of multi-level networks by the back propagation of error by Rumelhart et al. [RHW85]. Independently, David Parker produced his own version of back-propagation [Par85]. Kohonen has studied extensively both linear and non-linear associative networks, as well as self-organizing versions of these networks [Koh84]. In addition, there has been research into more abstracted networks such as those of Feldman and his group at the University of Rochester, the most notable of that work are the studies of connectionist vision and

evidential reasoning [ShF85]. A summary of connectionist and parallel distributed processing research is presented by Rumelhart and McClelland [RuM86]. Another important perspective on learning algorithms is coming from the neurobiological community as they develop more abstracted, "functional" models of biological structures. An excellent example of this work is that of Gary Lynch, Rick Granger, and their colleagues at the University of California at Irvine [Lyn86].

Connectionist/neural network researchers are learning to program networks that exhibit a broad range of cognitive behavior. Unfortunately, existing computer systems are limited in their ability to emulate such networks efficiently. Consequently, the Cognitive Architecture Project (CAP) at the Oregon Graduate Center is studying the implementation of special purpose VLSI architectures for the emulation of a very large neural networks.

The goal of CAP is to build an ULSI (Ultra-Large-Scale-Integrated[2]) neurocomputer. Such a system will be able to handle networks with thousands of nodes and millions of connections at rates faster than real time (biologically equivalent processing time). The manufacturing costs for such a system will be a few thousand dollars. It is our strong belief that such systems are necessary if neural networks are to be used in everyday applications.

The cost of emulating a network, whether with special purpose, highly parallel silicon-based architectures, or with traditional parallel architectures, is *directly* proportional to the number of connections in the network. This number tends to increase geometrically as the number of nodes increases. Even with large, massively parallel architectures such as ours, connections take time and silicon area (which is directly proportional to cost). Many existing neural network models scale up poorly, precluding large implementations. For

[2] Here we use ULSI to mean a ultra large silicon die size. Traditional microprocessor technology allows a die size of up to one square centimeter, beyond which, yield (the number of good dice per wafer) drops drastically due to inherent silicon fabrication faults, making a larger die size uneconomical. Due to the fault tolerance of neural networks, a significantly larger die should be cost-effective and may eventually encompass an entire wafer.

example, the number of connections in a Hopfield [Hop82] associative network scales up as the square of the number of network nodes. In its current formulation, this exponential increase in connections also holds for the Grossberg/Carpenter [CaG86] adaptive resonance network (ART). Most network models consist of either near total recursive connectivity (as with Hopfield) or a layered, feedforward network that has total connectivity of all nodes in a layer to all nodes in the next layer. As we scale networks to the large sizes required for real applications, these connectivity requirements significantly impact the implementation efficiency of the networks.

A second area of concern for self-programming networks is the scalability of learning time. A larger network has an even larger number of weights to program and a larger solution space to search. Unless some mechanism is added for controlling the solution space, the learning time increases exponentially too. For example, Barto [personal communication] believes that the learning time for stochastic learning automata [BSA83], as currently formulated, increases geometrically with the size of the network.

An example of this scalability problem can be seen by studying the back-propagation learning technique. Back-propagation networks generally are connected in a layered, feed-forward network, where each node in a level is connected (synapses with) every node in the succeeding level. When implementing back-propagation in VLSI, these feed-forward connections can be done efficiently using a broadcast technique (each node in the level broadcasts its values to all the nodes in the next level). However, when back propagating the error, point-to-point communication is required, since the error term for a particular node is the product of the forward weight and the error term of the forward node, which is unique for each connection [BHM]. Forward communication of signal can be done in $O(n)$ operations (where n is the number of nodes), but the backward communication of error in the worst case requires $O(n^2)$ operations!

There has been little study of these scaling issues and their impact on physical implementation. There was only a single paper at the 1987 International Conference on Neural Networks that dealt with connectivity issues: Somani and Penla [SoP87] addressed the performance of Hopfield networks with sparse, random connectivity. At the most recent AAAI conference Ballard discussed a modular back-propagation technique [Bal87].

We have chosen standard CMOS over more exotic technologies for implementing our architectures, because it has functional density, simplicity, reliability, and low cost, advantages that when taken together are unmatched by any other technology. In addition, CMOS is a mature, available technology that has benefited from the intense market pressures for memories and microprocessors.

CMOS VLSI has many characteristics that match well the characteristics of Connectionist/Neural Networks:

a) VLSI and neural networks are both asynchronous;

b) VLSI and neural networks are both massively parallel; and

c) VLSI is faulty, but neural networks are fault-tolerant.

But a major problem arises because:

d) VLSI is planar with 2 (or 3) levels of metal interconnect (we only consider metal here, since for the most part we are interested in long connections). These metal wires must observe certain inter-wire distance constraints that limit the total number of direct metal connections that can be made, thus creating an interconnection structure with a few high-bandwidth connections. Neural networks, on the other hand, have a large number of low-bandwidth connections, which is particularly true in associative processing where only a fraction of all connections are used at any one time. So even though neurons themselves are large compared to transistors, their axon and dendritic processes are much smaller than current or projected metal lines, and these processes

can be packed tightly in a three dimensional space. Taken together these characteristics constitute a serious mismatch between silicon and biological neural systems.

1.1. How to Solve the Connectivity Problem

The first decision that must be made when emulating large neural networks in VLSI is to determine the "virtualization granularity," that is, the degree of virtualization of network nodes and connections. A node or connection is said to *virtual* if it is multiplexed by physical hardware. In the *direct* or non-virtual approach, there is no sharing of hardware by multiple connections or nodes. The degree of sharing is the *virtualization granularity.*

The neural network to be emulated can be visualized as a large, multidimensional, directed graph, called the *c-graph.* The physical network that emulates the neural network can be visualized as a group of processing elements that signal each other through physical interconnection structures. This network also forms a graph structure called the *p-graph.* Because of the fine granularity of the c-graph, the processors (the nodes in the p-graph) may have a range of size or physical granularity. The c-graph is generally larger than the p-graph, and when the c-graph is mapped to the p-graph, a subset of the c-graph is emulated by each physical processor (a node in the p-graph).

At one end of the spectrum, we have one processor that emulates the entire c-graph, one connection at a time. Near, but not at this end of the spectrum are various multiprocessor implementations, such as the Intel iPSC hypercube system, and TRW ADAPT Mark III. At the other end of the spectrum are the "direct" implementations, where each physical processor emulates one node in the c-graph and each metal line is an edge in the c-graph. The neurochips built at Bell Labs and Carver Mead's silicon retina are examples of direct implementation. And of course one can imagine a large number of intermediate implementations between these two extremes.

One of our goals is to find the optimal point on this spectrum. The location of this point is highly application and technology dependent and is a result of the *communication locality* of the c-graph. Communication locality will be defined more formally below, but informally one can imagine a "virtual" communication network that consists of c-graph nodes called connection nodes or *CNs*. This network is mapped onto a physical network consisting of physical processors or *PNs* (physical processor nodes). In a spatially *local* connection network, most connections from one CN to another need only traverse a relatively small physical distance for a "good" mapping of the connection network onto a lower dimensional physical network of PNs. That is, most of a CN's connections are to CNs resident on the same PN, or to PNs that are local neighbors to its PN, with only a few connections to more distant CNs.

The spatial aspect of communication locality allows the construction of efficient busing structures. The temporal aspect allows the efficient multiplexing (or sharing) of those structures by a number of "virtual" connections. We have developed a variety of techniques for efficiently multiplexing scarce VLSI communication resources [BaH86] that take advantage of this locality. An important characteristic for efficient implementation is that the connection or virtual network exhibit spatial and temporal locality. Such locality is essential if the network is to scale to very large numbers of elements.

1.2. Biological Networks

There is much evidence that real neural networks also exhibit locality [Mou77]. Because of metabolic and genetic costs, it is likely that nature tends to use only as many connections as necessary for required functionality and reliability. There are a number of possible reasons for sparse connectivity[3]:

(1) Redundancy of input data. Sensory input data (in particular auditory and visual) is both spatially and temporally redundant. Since neural systems reflect the environment in which they operate, we expect that significant connectivity reduction results from the influence of such mutual information [McK85].

(2) Selective attention and limited short-term memory. Biological systems can only consciously attend to a few simultaneous stimuli, thus limiting the degree of simultaneous interconnectivity.

(3) Abstraction. Nervous systems have the ability (which is more pronounced in higher order mammals) to abstract or "chunk" groups of concepts into a new concept; this convergence of information performs an encoding that reduces inter-module bandwidth requirements [Bar85, Wic79].

(4) Functional Decomposition. Despite the regularity and distribution of processing, research has shown distinct functional areas within all nervous systems. The best example of this has been in the decomposition of the visual and auditory processing areas in primates [RoD85].

(5) Limited Degrees of Freedom. Most motor control systems have some restrictions to arbitrary degrees of freedom thus localizing control and feedback information by restricting the number of possible output combinations [ArS72].

An excellent example of constrained connectivity is in the visual subsystem in the monkey, from the retina through the lateral geniculate nucleus to the striate cortex, and then through the visual areas V1, V2 and V4 [DSM85]. These subunits are structured as levels of units, where each level consists of many parallel interacting cells. It appears (though it has not been proven) that each cell in a level has a limited receptive field of the cells from the previous level. These receptive fields appear to be fairly small in the lower levels of the network, slowly becoming more encompassing (directly and indirectly) as one

ascends the hierarchy. The presumed computational process is one of parallel hierarchical feature extraction, until at the upper level a few large, complex, and position independent structures are recognized. Several neural network models, such as those of Fukushima [FMI83,Fuk84], exhibit this type of structure. Linsker [Lin86a,Lin86b,Lin86c] has shown that given a Hebbian-like learning rule, multiple hierarchical layers of neurons with Gaussian distributed receptive fields will organize themselves into on-center, off-surround constructs in the lower levels and ocular dominance columns in the upper layers, similar to that of the visual system.

In this paper, a technique is presented for analyzing the effects of locality on the process of association. These networks use a more complex CN similar to the higher-order learning units of Maxwell et al. [MGL86a,MGL86b].

2. NETWORK MODEL

This section presents a model for and analysis of a *single* layer (row) of CNs (Connection Nodes), which may be a component of a much larger system. A technique is presented for analyzing the performance of a CN in a level versus its input connectivity. For the networks under consideration, this per level trade-off is independent of the general functionality of the network. The general approach of our research is to first develop a set of analytic tools in this restricted context, and then augment their capabilities to handle more general connectivity versus function issues.

Each such level in a multilevel network is fed by a previous level, and feeds a succeeding level. This model is similar to the multi-level networks of Rumelhart et al., Grossberg, and Fukushima. The inputs to a particular level may come only from the previous level as in feedforward networks, or they could also be recurrent or feedback

connections from the succeeding level or from other CNs in the same level. Figure 1 shows an example of a feedforward network.

Each CN has a number of functional sites requiring multiple inputs, where input values are combined together to produce a single site value, these values are then summed and passed through a non-linear sigmoid function to produce an output value. The functional sites that I have chosen correspond to the *r-codons* of Marr [Mar70]. R-codons, as will be discussed in more detail below, allow us to trade-off local computation for non-local communication by capturing information on higher-order correlations in the input space. Non-local (to other distant CNs) communication requirements are reduced and locality is increased when fewer, more "abstracted," units of information need be communicated. In other words, fewer "bits" are necessary if those that are communicated are used more efficiently.

One simplifying assumption I have made is that these networks are static and do not perform dynamic learning, that is, the connection weights are "compiled" once before the system is run. This assumption simplified the analysis, but will be abandoned in future

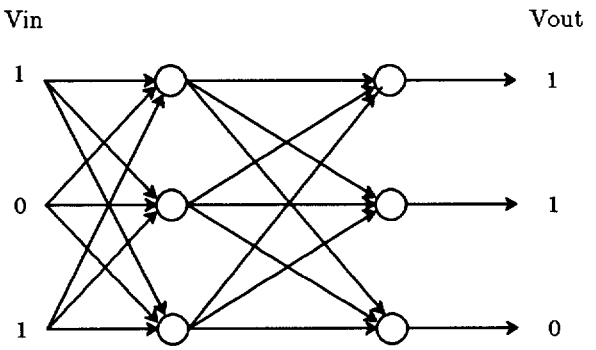

Figure 1 - A Multi-level Network with Feedforward Connections

research, since it is also important to understand the effect increased locality has on learning time.

A relaxation mechanism has been added which is similar to Marr's [Mar70]. To relax the network, a threshold value is input into each summation unit. There is one global threshold value for each level of CNs (an entire system in the case of auto-associative networks). The threshold is set to a value that maintains a constant number of CNs above threshold. There is much discussion in the literature about such competitive *noise suppression* or *quenching*. Marr postulates such a system in both the cerebellar and cerebral cortical models, and Grossberg's networks utilize sophisticated noise suppression mechanisms. This relaxation mechanism is a form of lateral inhibition, which provides the feedback essential to noise filtering. Grossberg has argued eloquently for competitive interaction in neural network models for noise suppression and code stabilization in dynamic networks. Any one familiar with the dynamics encountered in electronics and control engineering will feel intuitively comfortable with this notion.

Each level in the network is performing association on an input vector and producing an output vector:

Definition 1: A *level* (slab) of CNs takes an input vector from an input vector space, and generates an output vector into an output vector space. Each CN takes input from the input space and outputs an element of the output space, therefore a level is only a single row of CNs. Let N_I be the number of elements in an input vector, and N_O, be the number of CNs (each CN has one output) in the level. The elements of the vectors are assumed to be positive real values between 0 and 1 inclusive. □

Definition 2: There is a set of M_I, N_I-bit learned input vectors, $\vec{x_i} \in \Lambda_I$. Each vector consists of N_I elements, $x_{ij} \in \{0,1\}$, and where $1 \leq i \leq M_I$ and $1 \leq j \leq N_I$. Associated with each input vector is a probability, p_i, where $\sum_{i=1}^{M_I} p_i = 1$. There is also a set of M_O, N_O-bit learned

output vectors, $\vec{y_i} \in \Lambda_O$. Each vector consists of N_O elements, $y_{ij} \in \{0,1\}$, and where $1 \leq i \leq M_O$ and $1 \leq j \leq N_O$. □

Definition 3: The *distance*, $d_h[\vec{s}_1,\vec{s}_2]$ between two vectors, $\vec{s}_1$ and $\vec{s}_2$ of N elements, where each element, s_{ij}, is a positive real value, is defined as

$$d_h[\vec{s}_1,\vec{s}_2] = \frac{1}{N}\sum_{k=0}^{n-1} abs(s_{1,k} - s_{2,k}) \tag{1}$$

□

When the s_{ij} are single bits, then d_h is just the Hamming distance between the two vectors. More complex, "higher-order," or structured distance measures are possible, but this measure is sufficient for the purposes of this paper.

Definition 4: Association is a mapping, A, of elements in the set of learned input vectors, Λ_I to elements in the set of learned output vectors, Λ_O, i.e., $A: x \rightarrow y$, where $x \in \Lambda_I$, and $y \in \Lambda_O$. Associated with each learned vector x is a *generalization* region of vectors, Z, such that $A: z \rightarrow y$, $\forall\ z \in Z$. □

The appearance of an arbitrary input vector causes the system to generate the output vector associated with the learned vector in whose generalization region the input vector belongs. Though not necessarily true in practice, it is assumed here that the set of generalization regions covers the entire input vector space. Golden presents a complex, probabilistic form of this definition for general neural network functionality [Gol86].

Definition 5: An *equipotential* association is an associative mapping where the regions are contiguous and surround each learned vector. The region boundaries are equidistant from each learned vector, based on d_h, between all pairs of learned input vectors. The regions thus generated form polytopes surrounding the input learned vectors. □

This definition of association, where the input and output vectors spaces are different, is often called *heteroassociation.* It is also possible to define an association from a

vector space onto itself.

Definition 6: An *auto-associative* mapping is an association where $N_I=N_O$ and $\Lambda_I=\Lambda_O$. □

The general model of association presented here is just a simple mathematical mapping from a domain of objects (the input vector space) to a range of objects[3] (the output vector space). The important characteristic of this mapping is the existence of the generalization regions. The ability to generalize is an important abstraction, and one that is difficult to implement on traditional sequential computers, since it often requires a complete search of the domain of learned vectors. Expert systems also perform a similar kind of mapping, but due to their discrete nature, most rule systems (even those based on fuzzy logic) do not have the flexibility in setting the region boundaries as is possible with large, multi-level connection networks.

Definition 7: The *input noise*, Ω_I, is the expected d_h between an input vector and the intended vector. The *output noise*, Ω_O, is the expected distance between network output and the learned output vector associated with the closest learned input vector. The *information gain*, G_I, is

$$G_I \equiv -\log\left(\frac{\Omega_I}{\Omega_O}\right) \tag{2}$$

□

This definition of association says nothing about where the boundaries are placed. However, when dealing with a single level of CNs, equipotential association is assumed, since a single level of CNs can only implement a subset of all possible equipotential associations. Association as defined here encompasses most neural network and connectionist models.

[3] In many real networks the mapping is actually from a region in the domain to a different (usually smaller) region in the range — a "more-to-fewer" mapping. Though the distance must be reduced in all cases, it need not go to zero, some output noise is common in many neural network models.

Association, incidentally, can be defined independently of neural models; connectionist/neural networks are just highly parallel, fault tolerant implementations that associate in $O(1)$ time!

In multi-level learning networks, the input to output mappings and the boundaries between the regions can be complex. Also, these systems require many learning trials while the system explores and sets the boundaries. Simpler categorical learning such as that of Carpenter and Grossberg's ART (Adaptive Resonance Theory) [CaG86] network can learn the association boundaries much more quickly, since the boundaries are simpler. Fast adaptation is also a characteristic of the Nestor model [RCE82], which operates by large granular boundary movement.

Since most learning networks utilize association as defined here (even if it is a subpart of a more complex structure, such as in Grossberg's networks), it is used as the base functionality for the network model. Consequently, the connectivity results presented here are applicable to most other models.

The network model is now defined, a single CN is shown Figure 2.

Definition 8: A ***recursive neural network***, called a ***c-graph*** is a graph structure, $\Gamma(V,E,C)$, where

- There is a set of CNs, V, that can take a range of positive real values, v_i between 0 and 1. There are N_v in the set.
- There is a set of codons, E, that can take a range of positive real values, e_{ij} (for codon j of CN i), between 0 and 1. There are N_c codons dedicated to each CN (the output of each codon is only used by its local CN), so there are a total of $N_v N_c$ codons in the network. For simplicity, it is assumed that N_c is the same for each CN.

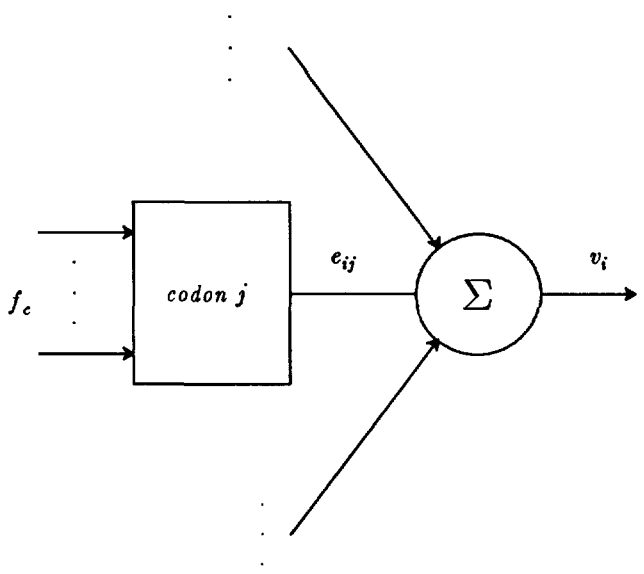

Figure 2 - A Single Connection Node (CN)

- $c_{ijk} \in C$ is a set of *connections*, of CNs to codons, $1 \leq i,k \leq N_v$, and $1 \leq j \leq N_c$, c_{ijk} can take two values {0,1} indicating the existence of a connection from CN k to codon j of CN i. □

The connection matrix, C, is three dimensional. The matrix, C', where $c'_{ik} = \bigcup_{j=1}^{N_c} c_{ijk}$ is two dimensional and provides general inter-CN connectivity information. □

Definition 9: The *CN receptive field* for CN i is the set of CNs from which CN i takes input, that is, the set k where $c'_{ik}=1$. The size of the CN receptive field for CN i is $f_v(i) = \sum_{k=1}^{N_v} c'_{ik}$. □

Definition 10: The *codon receptive field* for codon j of CN i consists of the set of CNs from which codon i,j takes input, that is, the set k where $c_{ijk}=1$. The *codon size* is the size of the codon receptive field for codon j of CN i is $f_c(i,j) = \sum_{k=1}^{N_v} c_{ijk}$. □

Minksy and Papert [MiP69] in their influential book on perceptrons defined the *order* and *diameter* of a perceptron. These definitions are similar to codon size and CN receptive field size respectively. In this paper it is assumed that all codon receptive field sizes are constant, that is, $\forall\ i,j,\ f_c=f_c(i,j)$, and that the CN receptive field sizes are constant, that is, $\forall\ i,\ f_v=f_v(i)$. Note, $1\leq f_c\leq f_v$ by definition. Later, a network interconnect cost measure will be defined over receptive fields that depends on a physical interpretation of network structure.

Definition 11: The value of CN i is

$$v_i = F\left(\theta+\sum_{j=1}^{N_c} e_{ij}\right) \tag{3}$$

The function, F, is a continuous non-linear, monotonic function, such as the sigmoid function. □

Network computation has been assumed to operate in the domain of real numbers. Later in this paper we will assume quantized input and output (generally binary), though real arithmetic is used internally. Silicon implementations will use either digital arithmetic, which can be defined as real arithmetic with quantization noise, or analog arithmetic, which is continuous, but is not exactly real arithmetic either.

F is a thresholding or "squishing" function. The purpose of the thresholding function is to add "gain" to the network elements. This gain is important in enhancing the noise filtering and decision making properties of the network.

In conjunction with the non-linear output function, a variable threshold can be defined that maximizes signal to noise:

Definition 12: The global network threshold θ is set to maximize the following objective function:

$$F\left[\sum_{i \in V_a} (v_i - 0.5 + \theta)\right] - F\left[\sum_{i \in V_{\bar{a}}} (0.5 - v_i + \theta)\right]. \quad (4)$$

Where V_a is the set of size a of CNs *st* $\forall\ v_i \in V_a$ and $\forall\ v_j \in V_{\bar{a}}$, $v_i \geq v_j$, and $V = V_a - V_{\bar{a}}$. □

That is, V_a is the subset of the set a of CNs with the largest outputs, where a is the *activation* level or duty cycle (the average number of activated CNs). Since the CN output is a sigmoid function, the above objective function always has a maximum for a given θ. The objective function specified here implies a centralized computation of θ, but, in multi-level networks, clusters of overlapping inhibitory regions can be used as a decentralized approximation.

A network with the activation level set to 1 is a classic *winner-take-all* network with a localized representation as discussed by Feldman and Ballard [FeB82]. When the activation level is set to some number greater than one, then a distributed representation [Hin84] results, "winners-take-all", where each CN may participate in more than one representation.

Definition 13: Define a mapping, $D(i,j,\vec{x}) \rightarrow \vec{y}$, where x is an N_I element input vector and $\vec{y}$ is the $f_c(i,j)$ element input vector of codon ij of CN i. Codon ij's output is e_{ij}. That is, $\vec{y}$ has as elements those elements of $\vec{x}$ where $c_{ijk}=1$, $\forall\ k$. □

Different input vectors may map to the same codon vectors, e.g., $D(i,j,\vec{x}) \rightarrow \vec{y}$ and $D(i,j,\vec{z}) \rightarrow \vec{y}$, where $x \neq z$. The codon values e_{ij} are determined as follows.

Definition 14: Let $\vec{x}(m)$ be input vector m of the M learned input vectors Λ_I for CN i. For codon ij of CN i, let T_{ij} be the set of f_c-dimensional vectors such that $\vec{t}_{ij}(m) \in T_{ij}$, and $D(i,j,\vec{x}(m)) \rightarrow \vec{t}_{ij}(m)$. That is, each vector, $t_{ij}(m)$ in the set of vectors T_{ij} consists of those elements of $\vec{x}(m)$ in codon ij's receptive field.

The variable l indexes the L vectors of T_{ij}. The number of distinct vectors in T_{ij} may be less than M, since, though the $\vec{x}(m)$ are distinct, the subsets, $\vec{\tau}_{ij}(m)$, need not be, since there is a possible many to one mapping of the $\vec{x}$ vectors of Λ_I onto each vector $\vec{\tau}$, that is, $L \leq M$.

Let $X^1 \in \Lambda_I$ be the subset of vectors in Λ_I, where v_i=1 (CN i is supposed to output a 1), and $X^0 \in \Lambda_I$ be those vectors where v_i=0, then define

$$n_{ij}^q(l) = size\_of\left\{D(i,j,\vec{x}_{ij}(m)) \; st \; v_i=q\right\} \tag{5}$$

for q=0,1, and $\forall$ m that map this l. That is, $n_{ij}^0(l)$ is the number of $\vec{x}$ vectors that map into $\vec{\tau}_{ij}(l)$ where v_i=0 and $n_{ij}^1(l)$ is the number of $\vec{x}$ vectors that map into $\vec{\tau}_{ij}(l)$, where v_i=1.

The ***information certainty*** of a codon for a vector $\vec{\tau}_{ij}(l)$ is defined as

$$HC_{ij}(l) = \frac{n_{ij}^1(l)}{n_{ij}^1(l)+n_{ij}^0(l)} \tag{6}$$

($HC_{ij}(l)\equiv 0$ when both n^1, n^0=0.) The output of codon ij, e_{ij}, is the maximum-likelihood decoding

$$e_{ij} = HC_{ij}(l'). \tag{7}$$

That is, where HC indicates the likelihood of v_i=1 when a vector $\vec{x}$ that maps to l' is input, and l' is that vector $\vec{\tau}(l')$ where $min\left[d_h(\vec{\tau}(l'),\vec{y})\right]$ $\forall$ l, $D(i,j,\vec{x})\rightarrow\vec{y}$, and $\vec{x}$ is the current input vector. In other words, l' is that vector (of the set of subset learned vectors that codon ij sees) that is closest (using distance measure d_h) to $\vec{y}$ (the subset input vector). There can be a different output e_{ij} for each compressed vector in codon ij's receptive field. □

The value of a codon is the "most-likely" output according to its inputs. For example, when there is no code compression at a codon, e_{ij}=1 exactly, if the closest (in d_h) subvector in the receptive field of the codon belongs to a learned vector where the CN is to output a 1 (and output a 0 if it is closest to a learned vector that requires a 0).

"Programming" a codon involves instructing it on the likelihood of the various input vectors it can see in its receptive field.

To help illustrate this information loss during code compression, an example is presented. Imagine the following mapping of learned input vectors to a CN, and the associated CN outputs:

x	CN input vector 012345	CN output
x(0) =	011010	v(0) = 0
x(1) =	101100	v(1) = 1
x(2) =	001100	v(2) = 1
x(3) =	000011	v(3) = 0

Assume that codon has a fan-in of 2 and takes as input bits 0 and 5 of the input field. The $\vec{t}$ and *HC* for the codon are

t(0) = 00	maps x(0) and x(2)	HC(0) = .5
t(1) = 01	maps x(3)	HC(1) = 0
t(2) = 10	maps x(1)	HC(2) = 1
t(3) = 11	no x	HC(3) = 0

The codons implement nearest-neighbor classification, which has an error rate that has been shown by Cover and Hart [CoH67] to be, for an arbitrary number of categories, within a factor of 2 of the ideal Bayesian error rate. Bayesian classification could be used, but simulations showed only a small performance improvement.

In this paper it is assumed that the vector mapping for each codon is computed statically prior to network operation. Marr's model for the cerebral cortex (upon which much of this model is based) performed dynamic learning (feature extraction) of the input vector set. It did so by extracting significant information from the learned vector set. It is clear that future connectivity analysis must contend with dynamic learning.

It is not the intent of this paper to propose that codon networks as described here should become the basis of connectionist and neural networks. The importance of this model derives from its mathematical properties that make it amenable to communication theoretic analysis. The behavior predicted by these models can then be approximated arbitrarily closely by more traditional higher-order networks.

To measure cost/performance, measures of cost and of performance are required. Cost is defined in section 3.3. For a measure of performance, network capacity is used.

Definition 15: The *capacity* of a network is the maximum number of learned vector to output vector mappings such that the information gain, G_I, is strictly positive (>0). □

The capacity of the network depends on a variety of factors and parameters such as the locality, number of codons, and receptive field size. There are several possible performance measures. They are network fault-tolerance, settling time, and capacity. Fault-tolerance can be determined by deleting random connections and examining at the resulting capacity. Consequently, fault-tolerance, though a crucial characteristic of connection networks, is not considered further in this report; it will be covered in more detail in future reports. The time required for the network to stabilize tends to be constant and independent of the size of the network, therefore, in the remainder of this paper, I will measure network performance solely on the basis of network capacity.

3. COMMUNICATION ANALOGY

Given the simple network model (for a single level) described in the previous section, the next question is how does one measure the trade-off between connectivity and capacity? What are the best values for the network parameters to maximize network performance in the face of sparse interconnect? To answer this question analytically, we need a model of

the association process. Since the problem of sparse interconnect is one of information transfer between different elements, it seemed natural to use communication theoretic techniques to analyze network performance.

The analysis of neural networks the last few years has proceeded in two directions simultaneously. Two approaches are the *structured,* and the *unstructured.* The grandfathers of structured analyses were McCulloch and Pitts [McC65], who analyzed neural networks as logical computing devices, and although some interesting theory resulted, their analytic tools were inadequate for handling the complexities of the nervous system. One reason for this is that neural function is the result of the mass action of large groups of neurons, creating emergent functionality that is not amenable to highly structured analysis.

More recently, unstructured approaches have been proposed that consider networks as amorphous groups of identical elements and use statistical theories such as the thermodynamics of spin glasses to understand collective functionality. Hopfield's work [Hop82] is a classic example of this approach. However, researchers have begun to realize that real neural systems may be more structured than the thermodynamic approaches can handle. Feldman has noted that what is needed are theories that are not amorphous like the gas models, but are more structured like those of molecular biology.

Consequently, the search is on for that middle ground. The Boltzmann machine of Hinton, Sejnowski, and Ackley [HSA84] and the Harmony theory of Smolensky [Smo86] are attempts to apply thermodynamic principles to more structured systems. The work of Grossberg [Gro86] and Kohonen [Koh84] is not statistical in nature, but they do successfully analyze large groups of cooperating neural elements in restricted domains. In this paper, another analysis technique, Communication Theory, is applied to neural networks. This theory is fundamentally probabilistic, but does deal with the structured aspects of these systems. It is not intended as "the" ultimate tool for neural network analysis, but rather as

another tool in the growing arsenal of techniques and strategies for describing and coping with neural system function. The main goal of this paper is to propose communication theory as a viable technique for dealing with fundamental issues of network interconnectivity.

Consider a single CN. Assume that the CN's output value space, contains two values, 0 and 1. Therefore, the CN must decide whether the input it sees belongs to the class of "0" codes, those codes for which it remains off, or the class of "1" codes, those codes for which it becomes active. The inputs it sees constitute a subset of the input vectors for the level of CNs to which the CN belongs. It is also assumed that the CN is an *ideal* 1-NN (Nearest Neighbor) classifier or feature detector, that is, given a particular set of learned vectors, the CN will classify an arbitrary input according to the class of the nearest (using d_h as a measure of distance) learned vector (as in equipotential association). This situation is equivalent to the case where there is a single CN that has a single codon whose receptive field size equals that of the CN ($f_c=f_v$).

Imagine a sender who wishes to send one bit of information over a noisy channel. The sender has a probabilistic encoder that choses a code word (learned vector) according to some probability distribution. The receiver knows this code set, though it has no knowledge of which bit is being sent. Noise then is added to the code word during its transmission over the channel. This operation is analogous to applying a vector to a network's inputs, where the vector lies within some learned vector's region. This "noise" is represented by the distance (d_h) between the input vector and the associated learned vector.

The code word sent over the channel consists of those bits that are seen in the receptive field of the CN being modeled. In the associative mapping of input vectors to output vectors, each CN must respond with the appropriate output (0 or 1) for the associated learned output vector. Therefore, the CN as defined above is a decoder that estimates in

which class the received code word belongs[4]. This is a classic block encoding problem, where increasing the field size is equivalent to increasing code length. And, as the receptive field size increases, the performance of the decoder improves in the presence of noise. Using communication theory then, the trade-off between interconnection costs as they relate to field size and the functionality of a CN as it relates to the correctness of its decision making process (output errors) can be characterized.

As the receptive field size of a CN increases, so does the redundancy of the input, though this is dependent on the particular codes being used for the learned vectors, since there are situations where increasing the field size provides no additional information. It will be shown below that for randomly selected codes, there is a point of diminishing returns, where each additional bit provides ever less reduction in output error. And, for multi-codon CNs, the error can actually increase in some situations by adding connectivity. Another factor is that interconnection costs increase exponentially with field size. The result of these two trends is a cost performance measure that has a single global maximum value. In other words, given a set of learned vectors and their probabilities, and a set of interconnection costs, a "best" receptive field size can be determined.

The rest of this section is devoted to the analysis of a single CN. In the first subsection, a communication model is developed and communication theory applied to a CN with a single codon with no code compression.

[4] For each CN, the intended output (0 or 1) can be thought of as the original bit that was to be sent over the channel, where the input learned vector represents an encoding of that bit.

3.1. Single Codon, With No Code Compression

In this section it is shown that a single neural element with a single codon and with no code compression can be modelled exactly as a communication channel. Each network CN is assumed to have a single codon whose receptive field size is equal to that of the receptive field size of the CN, i.e., $f_c=f_v$. This codon operates as a 1-NN classifier.

Figure 3 shows a communication channel. The operation of the channel is as follows. A bit is input into the channel encoder, which selects a random code of length N and transmits that code over the channel. The receiver then, using nearest neighbor classification, decides if the original message was either a 0 or a 1.

Let M be the number of code words used by the encoder, which corresponds to M_I; M_O is always 2 for a single CN (assuming binary output). The rate[5] then indicates the density of the code space.

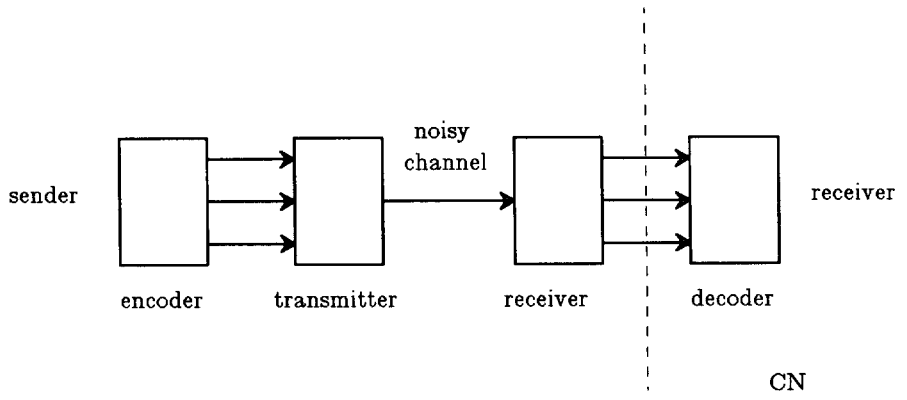

Figure 3 - A Communication Channel

[5] In the definitions given here, and the theorems below the notation of Gallager [Gal68] is used. Many of the definitions and theorems are also from Gallager.

Definition 16: The *rate*, R, of a communication channel is

$$R \equiv \frac{\log M_I}{N_I} \tag{8}$$

□

The derivations in later sections use a related measure:

Definition 17: The *code utilization*, b, is the number of learned vectors assigned to a particular code or

$$b \equiv \frac{M_I}{2^{N_I}} \tag{9}$$

b is essentially unbounded, since M_I may be significantly larger than 2^{N_I}. When $b \leq 1$ then there is no code compression, likewise when $b > 1$, then code compression occurs that causes information loss. For single codon analysis, $N_I = f_c$, the receptive field size of the codon. □

In the analysis of single CN performance, a binary output is assumed. All logarithms are to the base 2 ($\log_2$). The reader should note that all theory developed here can be extended to values of any discrete granularity.

When modelling a codon as a communication channel, then the maximum rate, R, is not related to the actual information content of the channel (which is at most one bit), but to the density of the code space. For this reason, the channel error (as is shown below) must be adjusted accordingly. The actual code word sent over the channel is the subset of the selected learned vector that the particular codon in question sees in its receptive field. The performance of the codon is directly related to any information loss that occurs due to code compression. (Code compression has a similar effect to exceeding the capacity of the transmission channel.)

The error due to code compression is a random variable that depends on the compression rate and the a priori probabilities, therefore, it will be different with different learned vector sets and codons within a set. As code utilization approaches and becomes

greater than 1, code compression occurs more often and decode error is unavoidable.

Increasing the block size generally increases the decoder's performance. However, when the channel capacity is exceeded, increasing block size does not help. When the code utilization increases, which is what happens when the number of learned vectors increases, the effect is similar, though the analysis is somewhat different. The remainder of this subsection assumes that there is no compression, the next subsection then examines CN performance with compression.

Let $\vec{x}_i$ be the vector output of the encoder, and the input to the channel, where each element of $\vec{x}_i$ is either a 1 or 0. Let $\vec{y}_i$ be the vector output of the channel, and the input to the decoder, where each element is either a 1 or a 0.

In a real communication channel, the code word is sent sequentially one bit at a time. In the CN model, the CN's receptive field receives the code word in parallel, but it is analyzed as if it had been received serially, by assuming a memoryless channel.

Definition 18: A memoryless channel is assumed, therefore, the ***channel transition probabilities***, that is, the probability of receiving a vector $\vec{y}$ given that the mth code word was transmitted, is

$$P_N(\vec{y}/\vec{x}_m) = \prod_{n=1}^{N} P(y_n/x_{m,n}) \tag{10}$$

□

The channel noise in the system is incorporated into the transition probabilities.

Definition 19: The ***maximum likelihood*** decoding rule is where, given $\vec{y}$, chose m' for which

$$P_N(\vec{y}/\vec{x}_{m'}) \geq P_N(\vec{y}/\vec{x}_m) \quad \forall \; m \neq m' \tag{11}$$

□

A decoding rule is a mapping (association) from the set of $\vec{y}$ vectors into a set of output codes. In the case of a CN, there are only two output codes, 0 or 1. When the source produces code word m, $\vec{x}_m$ is transmitted, then an error occurs if the received code is in the set of vectors $\vec{y}$ that the decoder does *not* map to m.

Definition 20: If the set of $\vec{y}$ that map to m is Y_m, then the set of vectors that do not map to m is the complement Y_m^C. Given that message m was sent, the probability of a decoding error is defined as

$$P'_{e,m} = \sum_{\vec{y} \in Y_m^c} P_N(\vec{y}/\vec{x}_m) \tag{12}$$

If the probability that m is sent is p_m, then the probability of a decoding error can be defined as

$$P'_e = \sum_{m=1}^{M} p_m P'_{e,m} \tag{13}$$

□

Because of the difficulty of computing these bounds for specific codes, the noisy channel communication theorem was proved for randomly selected codes [Gal68, ShW49]. For a random code of length N, the bits of the code are chosen randomly. Consequently, the bounds shown here are for randomly chosen codes rather than optimally selected codes. This means that there are codes that do better and codes that do worse than the stated bound. The codes encountered in connectionist and neural systems, though containing structure, will have random tendencies, so the assumption of randomly chosen codes is more realistic here than in communication theory where codes are often hand optimized for a particular application.

Definition 21: If each bit of the code is chosen independently, then the overall probability of a single code word, $\vec{x}$, of length N is defined by

$$Q_N(\vec{x}) = \prod_{n=1}^{N} Q(x_n) \tag{14}$$

where $Q(i)$ is the single bit probability assignment. □

The Noisy Channel Coding Theorem is now presented for the general case where the individual M input codes are to be distinguished. The result is then extended to a CN, where, even though M input codes are used, the CN need only distinguish those codes where it must output a 1 and those where it must output a 0. The first theorem is from Gallager (5.6.1) [Gal68]:

Theorem 1: Let $P_N(\vec{y}/\vec{x})$ be the transition probability assignment for sequences of length $N \geq 1$ on a discrete channel. Let $Q_N(\vec{x})$ be an arbitrary probability assignment on the input sequences and, for a given number, $M \geq 2$, of code words of block length N, consider the ensemble of codes in which each word is selected independently with the probability measure $Q_N(\vec{x})$. Suppose that an arbitrary message m, $1 \leq m \leq M$ enters the encoder and that maximum-likelihood decoding is employed. Then the average probability of a decoding error over this ensemble of codes is bounded, for any choice of ρ, $0 \leq \rho \leq 1$, by

$$P'_{e,m} \leq (M-1)^{\rho} \sum_{\vec{y}} \left[\sum_{\vec{x}} Q_N(\vec{x}) P_N(\vec{y}/\vec{x})^{\frac{1}{(1+\rho)}} \right]^{1+\rho} \tag{15}$$

□

The above theorem can be rewritten by considering (N,R) block codes where $M-1 < 2^{NR} \leq M$. The theorem and proof are also from Gallager (5.6.2) [Gal68].

Theorem 2: Let a discrete memoryless channel have transition probabilities $P_N(j/k)$ and, for any positive integer N and positive number R, consider the ensemble of (N,R) block codes in which each letter of each code word is independently selected according to the probability assignment $Q(k)$. Then, for each message m, $1 \leq m \leq \left\lceil e^{NR} \right\rceil$, and all ρ, $0 \leq \rho \leq 1$, the ensemble average probability of decoding error using maximum-likelihood

decoding satisfies

$$P'_{e,m} \leq \exp\left\{-N\left[E_0(\rho,Q)-\rho R\right]\right\} \tag{16}$$

where

$$E_0(\rho,Q) = -\ln \sum_{j=0}^{J-1} \left[\sum_{k=0}^{K-1} Q(k) P(j/k)^{\frac{1}{1+\rho}} \right]^{1+\rho} \tag{17}$$

□

These results are now adjusted for our special case.

Theorem 3: For a single CN, the average channel error rate for random code vectors is

$$P_e \leq 2q(1-q)P'_{e,m} \tag{18}$$

where $q=Q(k) \; \forall \;\; k$ is the probability of a 1 bit.

Proof: The probability that the CN is to output a 1 is q. The probability that an error occurred is $P'_{e,m}$. However, an error moves the decoder to another random output, which has probability q of still giving the correct output, and $1-q$ of incorrect output. Therefore, the probability of an error, when the correct output is 0 and 1, is:

$$q(1-q)P'_{e,m}+(1-q)qP'_{e,m} \tag{19}$$

□

For q=0.5, this reduces to $P_e \leq 0.5P'_{e,m}$.

The results given thus far cover a wide range of communication channel models. A more easily computable expression can be derived by recognizing some of the restrictions inherent in the CN model. Assume that all channel code bits are equally likely[6], that is, $Q(k)=q, \forall \; k$, that the error model is the BSC (Binary Symmetric Channel), see Figure 4,

[6] This assumption violates the assumption of structure in the input data stream. The incorporation of such higher order structure is beyond the scope of this paper.

and that the errors are identically distributed and independent, where each bit has the same probability, ϵ, of being in error (a 0 to a 1 transition or a 1 to a 0 transition) independent of its position in the code word, and of the code word itself. With these restrictions the relationships described by Theorem 3 can be simplified:

The bound of theorem 2 can be simplified by letting, $Q(0)=q=0.5$, $Q(1)=1-q=0.5$, $P(0/0)=1-\epsilon$, $P(0/1)=\epsilon$, $P(1/0)=\epsilon$, $P(1/1)=1-\epsilon$. Maximizing ρ gives the tightest bounds:

$$P_e \leq \max_{0\leq\rho\leq 1} P'_e(\rho) \tag{20}$$

For the BSC assumed here, the minimum value for equation 17 is obtained when $\rho=1$, and

$$E_0 = -\log 2 \left[\left(0.5\sqrt{\epsilon}+0.5\sqrt{1-\epsilon} \right)^2 \right] \tag{21}$$

In this section it was assumed that a CN had a single codon. From this assumption, results from communication theory could be applied. Though it is useful as an absolute upper bound on CN performance, modifications are necessary to extend this analysis to more realistic CNs that have larger numbers of smaller codons.

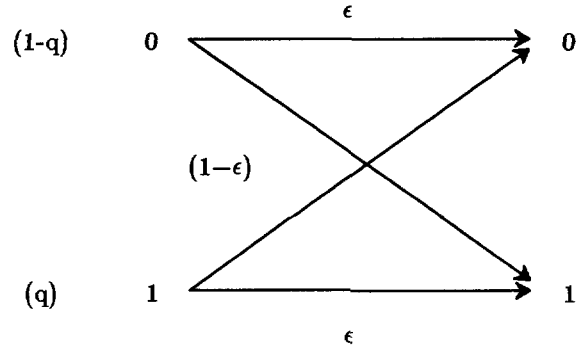

Figure 4 - A Binary Symmetric Channel

Before turning to the analysis of a CN with multiple codons, the effect that compression (aliasing due to limited receptive field size) has on the error bounds derived above are characterized.

3.2. Single-Codon with Code Compression

The implementation complexity of a codon grows exponentially with the size of the codon, which limits its practical size. An alternative is to approximate the single codon function of a single CN with many small, overlapped codons, improving the cost/performance of the decoding process. As codons get smaller, their receptive field size becomes smaller relative to the number of CNs in the network. When this happens there is code compression, or *vector aliasing*, that introduces error into the decoding process. Networks can overcome this error by using multiple redundant codons with overlapping receptive fields.

In this section the error rate of a single codon is studied as its receptive field size is reduced (while maintaining the vector size, N, and the number of learned vectors, M, constant). The next section then examines the behavior of a group of multiple small codons in a single CN, and compares that CN's error rate to the ideal CN with a single large codon.

Compression occurs when two code words requiring different decoder output share the same representation (within the receptive field of the codon). Recall that $HC_{ij}(l)$ is the information certainty for vector l at codon j of CN i:

$$HC_{ij}(l) = \frac{n_{ij}^1(l)}{n_{ij}^1(l)+n_{ij}^0(l)} \tag{22}$$

The learned vectors are random, hence for any single l, the $n^1(l)$ (dropping the ij subscripts for convenience) are random variables that can be modelled as a Bernoulli sum of a set of binary trials. Recall that the receptive field size of the codon is f_c. One can

imagine then that there are 2^{f_c} containers, one for each vector that the codon could potentially respond to, and that there are M_I tokens, one for each learned vector. There are two kinds of tokens: 1 and 0. The 1-tokens correspond to those vectors where the codon is supposed to output a 1 and the 0-tokens to those that require a 0. A 1-token is placed into a certain container when the subfield, $D_{ij}(l)$, of a learned vector matches that of a certain container. So, n^1 is the number of 1-tokens, and n^0 the number of 0-tokens.

Since vectors are selected randomly, the number of tokens of each type in a container is a random variable and corresponds to a sequence of Bernoulli trials, where each token is a binary sample. The 1-tokens are chosen according to probability q, the 0-tokens to probability, $1-q$. Assume that all containers are uniform, i.e., $n^1(l)=n^1, \forall\ l$, where the number of balls in the container is just the average code utilization, b, which is suitable when M is larger than f_c. For the remainder of this paper, b as defined here is used as an estimate for the degree of code compression. The actual b may vary slightly from container to container and hence some code compression is possible even when $b<1$.

On the average when the code utilization, b, is >1, then there is code compression and the bounds of the previous section do not apply. Likewise, when $b<1$, there is little or no compression and the bounds derived here do not apply.

Define $p_1(k)$ as the probability that there will be k 1-tokens out of the b total tokens in the container,

$$p_1(k) = \binom{b}{k} q^k (1-q)^{b-k} \tag{23}$$

for any k, $0 \leq k \leq b$.

When a vector is received, the codon operates as in the case without compression by finding the closest, "best-match" (according to the measure of Definition 3), non-empty "container" that matches the vector field it sees. The codon then outputs the *HC* (ratio of 1 tokens to all tokens in the container) associated with the container. If there is a single

codon, then the CN output is 1 if $HC \geq 0.5$ and 0 if $HC < 0.5$.

If the number of samples is large, then a normal distribution can be used as an approximation to a binomial distribution for $p_1(k)$. In addition, the decoder is given one extra bit of information, it knows the type (1 or 0) of the majority tokens in the container, since in the networks defined in this paper, the codon itself has that information. This means that the codon, though not always correct, is correct the majority of the time. If it always assumes the majority carrier, in the long run, the percentage of incorrect guesses will be equal to the number of minority tokens in the container. Consequently, every time a vector is decoded by the codon, chose a p_c (according to a half Gaussian distribution, Figure 5), which is the decode error due to code compression at the codon, corresponding to the random selection of a container. An expression for the mean compression error, $\bar{p}_c$, is now derived.

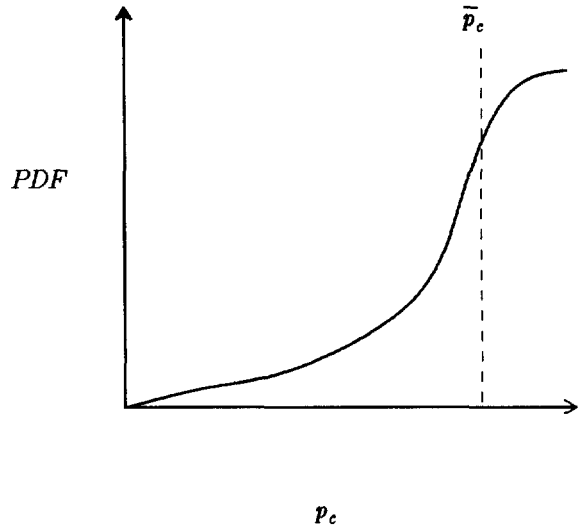

Figure 5 - Half Gaussian Approximation

Let $B_k(b,q)$ be the expression for the probability that k 1-tokens have occurred in a group of b tokens, or

$$B_k(b,p) = \binom{b}{k} q^k (1-q)^{b-k} \tag{24}$$

Define $E_k(b)$ as the error that occurs due to compression, if there are k 1-tokens in a container, or

$$E_k(b) = \begin{cases} \frac{k}{b} & \text{if } k \leq \frac{b}{2} \\ 1-\frac{k}{b} & \text{if } k > \frac{b}{2} \end{cases} \tag{25}$$

Then

$$\bar{p}_c = \sum_{k=0}^{k=b} E_k(b) B_k(b,q) \tag{26}$$

or,

$$\bar{p}_c = 2 \sum_{k=\frac{b}{2}}^{k=b} (1-\frac{k}{b}) B_k(b,q) \tag{27}$$

By changing the indices, a normal approximation to the binomial distribution, $B_k(b,q)$ can be used to obtain:

$$\bar{p}_c = \frac{2}{\sqrt{2\pi}} \int_0^{\frac{b}{2}\sigma} (.5-\frac{y\sigma}{b}) e^{-1/2y^2} dy \tag{28}$$

where $\sigma = \sqrt{bq(1-q)}$. A closed form solution is possible, by using the following identity

$$\frac{1}{2f} = \int_0^{\infty} y e^{-fy^2} dy \tag{29}$$

giving

$$\bar{p}_c = 0.5 - \frac{2\sqrt{bq(1-q)}}{b\sqrt{2\pi}} \tag{30}$$

The codon first decodes the vector and finds the most-likely container, it then determines the error due to compression.

Theorem 4: For a BSC model where q=0.5, the codon receptive field is f_c, the code utilization is b, and the channel bits are selected randomly and independently, the probability of a codon decoding error when $b>1$ is approximately

$$P_{cdn} \leq (1-\epsilon)^{f_c}\ \bar{p}_c - \left[(1-(1-\epsilon)^{f_c})\right]0.5 \tag{31}$$

and when $b<1$

$$P_{cdn} \leq \exp\left\{-f_c\left[-\log 2\ \left(\left[(0.5\sqrt{\epsilon}+0.5\sqrt{1-\epsilon}\right]^2\right)-R\right]\right\} \tag{32}$$

Proof: The proof follows from the bounds in the previous section, and the fact that if all code words are used ($b>1$), then the decode error for a single word is $(1-\epsilon)^{f_c}$. □

These bounds are approximate, but as the simulations discussed below indicate, they are reasonably predictive. As b gets large, the variance gets small and $\bar{p}_c$ approaches 0.5 asymptotically. It is obvious that the performance of a single codon degrades rapidly in the presence of even small amounts of compression. For this reason, I now turn to the use of multiple small codons within a single CN as a means to overcome the error, that is due to compression, of a single codon.

3.3. Multiple Codons with Code Compression

The use of multiple small codons is more efficient than a few large codons, but there are some fundamental performance constraints. When a codon is split into two or more smaller codons (and the original receptive field is subdivided accordingly), there are several effects to be considered. First, the error rate of each new codon increases due to a decrease in receptive field size (the codon's block code length). This change increases the error rate per codon.

The second effect is that the code utilization, b, will increase for each codon, since the same number of learned vectors is mapped into a smaller receptive field. This change also increases the error rate per codon. As the individual codon receptive fields get smaller, this error increases rapidly. For higher-order input codes, there is an added error that occurs when the order of the individual codons is decreased. Such higher-order information can be captured by adding multiple levels, but the analysis of that is beyond the scope of this study.

The third effect is the mass action of large numbers of codons. Even though individual codons may be in error, if the majority are correct, then the CN will have correct output. This change decreases the total error rate. Key questions then are, How many codons are need to maintain a given CN decode error rate as the size and complexity of each codon is reduced?, and, What are the cost-performance trade-offs?

Assume that each CN has some ordinal number of codons, c, where $c>1$. The union of the receptive fields for these codons is the receptive field for the CN. There are no restrictions on the degree of overlap of the various codon receptive fields within or between CNs. For a CN with a large number of codons such overlap will generally be random and uniformly distributed. I am assuming that the transmission errors seen by the receptive fields are independent. This is somewhat unrealistic, but it significantly simplifies the computation of CN error; for small error rates and large numbers of codons it is a reasonable assumption,

I am interested in what happens to the codon decode error due to compression (ignoring transmission error for the time being) when a codon is replaced by two or more smaller codons covering the same receptive field. This process can continue until there are only 1-codons, which, incidentally, is analogous to most current neural models. For a multiple codon CN, assume that each codon votes a 1 or 0. The summation unit then totals this

information and outputs a 1 if the majority of codons vote for a 1, etc. The network model presented in section 2 uses the information uncertainty, HC as the codon output, to simplify analysis, this graded codon output is ignored. The probability of voting is p_c, which is itself a random variable. The decode operation can be modelled as follows. First, each codon is sampled and a p_c selected according to the half-Gaussian distribution of the previous section. Then a token is selected from two types, "error" and "correct," according to the probability p_c and placed into a container. The CN's output is correct if the majority of the tokens are correct.

To simplify the analysis further, use the mean $\overline{p}_c$ as computed in the previous section, which assumes a large amount of compression, and a large number of samples, therefore, $\overline{p}_c$ will be close to q, and there will be only a small variance around q. Once again this is a Bernoulli sample of a binary event, and since c will generally be large, it can be approximated with a gaussian distribution. The probability of a CN error due to compression then is just

$$P_c = \frac{1}{\sqrt{2\pi}} \int_{\frac{c/2-c\overline{p}_c-1/2}{\sqrt{c\overline{p}_c(1-\overline{p}_c)}}}^{\infty} e^{-\frac{1}{2}y^2} dy \tag{33}$$

P_c incorporates the last two effects of moving to multiple smaller codons. Substituting into the equation given in the previous section gives the total error probability (per bit), $P_{CN}(f_c)$:

$$P_{CN}(f_c) = P'_e(f_c)+P_c-P'_e(f_c)P_c \tag{34}$$

where $P'_e(f_c)$ is the single codon (no compression) bound per codon. The receptive field size per codon (not per CN) is f_c.

As is shown in the next section. the cost of a codon increases exponentially with its receptive field size, therefore, smaller codons are preferred. However, things look dismal for the small codon, since P_c rapidly approaches q, where the codon provides no information to

the CN, thus reducing information gain. This effect can be offset to some degree by adding ever more codons, but there are other problems.

Consequently, for networks that perform association as defined in this paper, the connection weights rapidly approach a single uniform value as the size of the network grows. In information theoretic terms, the information content of those weights approaches zero as the compression increases. Why then do simple non-conjunctive networks (1-codon equivalent) work at all? Since one of the main goals of this paper is to understand the behavior of associative network models as the size of the network increases, this is an important issue. Nervous systems obviously scale, so what architectural characteristics allow them to overcome the problems stated here?

In the next section I define connectivity cost constraints and show that the answer to the first question is that the general associative structures defined here *do not* scale cost-effectively and more importantly that there are limits to the degree of distribution of information. The answer to the second question will be discussed near the end of this paper, and a detailed treatment will be given in follow on papers. These architectural characteristics are the result of the fundamental information theoretic bounds that constrain network connectivity patterns.

3.4. Connectivity Costs

Our goal is to build a physical computer system that emulates these networks, therefore, it is important to characterize the trade-offs between network performance and cost, in particular as it relates to the parameters that be controlled, such as CN and codon receptive field size, codon size, and numbers of codons.

The cost measure is directly related to the physical implementation where the connection network, Γ, is mapped to an array of physical processors configured into a 2-dimensional grid.

Definition 22: A processor connectivity graph or ***p-graph***, $\Theta(U,C_P)$, consists of a set of N_P processors, U, and a *directed* interconnection graph described by C_P. □

The p-graph describes the interconnection structure that the processors in the grid (of $\sqrt{N_P}$ by $\sqrt{N_P}$ elements) use to communication with one another. We have defined the p-graph as being directed, when, in many implementations, physical interconnection structures are generally bidirectional. However, for most technologies, it is much more expensive in space and time to turn around physically long interconnect. Also, symmetrical connectivity can be approximated by letting $C_P = C_p^T$, that is, for every connection from PN_i to PN_j, there is a connection from PN_j to PN_i.

Definition 23: The ***c-graph distance***, $d_c(C,i,j)$, between two CNs, v_i and v_j in the connection graph, C, is the number of *directed* edges in the Network graph that must be traversed to reach i from j. □

Definition 24: Define a mapping $B{:}\Gamma \rightarrow \Theta$, which maps the CNs of the c-graph, Γ, to the processor array, Θ. This is a possibly many to one mapping of CNs to PNs, and of the edges in the c-graph to paths of the p-graph. □

The c-graph then is a *virtual* network that is emulated by a physical network. For the research presented here, a simple one to one mapping of CNs to PNs is assumed, though many real implementations will generally map multiple CNs per PN [BaH86].

Definition 25: The ***p-graph mapped distance***, $d_p(C,B,i,j)$, between two c-graph CNs, v_i and v_j for a mapping B is the number of p-graph edges required when following the directed path from the PN where CN i resides to the PN where CN j resides. □

Definition 26: Given a graph, $\Gamma(V,C,E)$, $r_i(n)$ is the number of CNs reachable in exactly n edge traversals from CN i. The *average reachability* function, $R_B(n)$ for the graph then is

$$R_B(n) = \frac{1}{N} \sum_{i=0}^{N-1} \left(\sum_{j=0}^{n} r_i(j) \right) \tag{35}$$

where N is the total number of CNs in the graph. □

Definition 27: The *locality*, $L(n)$, of a connection graph, Γ, is the inverse function of R, written R^{-1}. In other words, $L_\Gamma(n)$ is the probability of a connection to a CN a distance, n interprocessor hops away:

$$L_B(n) = \frac{1}{N} \sum_{j=0}^{N-1} \frac{r_j(n)}{\left(\sum_{i=0}^{N-1} r_j(i) \right)} \tag{36}$$

□

A mapping independent definition of locality is possible, by mapping the c-graph onto itself.

It is much easier to assess costs if some implementation medium is assumed. I have chosen standard silicon, since it is relatively straightforward to estimate costs and since it is our preferred technology. Silicon provides a two dimensional surface where CN's and codons take up area according to the size of their receptive fields. In addition, there is area devoted to metal lines that interconnect the CNs. A specific VLSI technology need not be assumed, since the comparisons are all relative, thus keeping CNs, codons, and metal in the proper proportions, according to a standard metal width, m_w (which also includes the inter-metal pitch). It is assumed that m_l levels[7] of metal are possible.

[7] In an aggressive state of the art CMOS processes, m_w is about 2 micrometers and m_l is 3.

Multi-dimensional implementation costs, as would be found in optical or 3-dimensional VLSI, could easily be factored into these analyses, but I have chosen 2-dimensional silicon, since that is the initial implementation technology chosen by the Cognitive Architecture Project for reasons of ease of use, low cost, flexibility, and availability.

Although the implementations being considered by the Cognitive Architecture Project assume the sharing of most metal lines by several "virtual" connections via multiplexing, it is assumed in this analysis that each metal line is dedicated to a connection, where there is one PN per CN and that the p-graph and c-graph are isomorphic.

In the previous section I established the relationship of network performance in terms of the transmission error rate, ϵ, and the network capacity, M. In this section I derive an implementation cost, which is total silicon area, A. A can then be used to derive a cost/performance figure that can be traded off with such factors as codon size, receptive field size, etc. Transmission delay is not factored into the performance figures, since increasing metal length increases inter-PN interconnect area and is reflected in A, and performance is being measured[8] only in terms of network capacity.

There are two components to the total area: A_{CN}, the area for each CN, and A_{MI}, the area of the metal interconnect between CNs per CN. A_{CN} only considers the area for the CN logic. The metal area for local, intra-CN interconnect is considered to be much smaller than that of the codon processing hardware. The area per CN then is roughly

$$A_{CN} = n_{CN} m_c K \tag{37}$$

where n_{CN} is the receptive field size for the CN, and m_c the number of vectors that each codon can distinguish, for $b \geq 1$, $m_c = 2^{f_c}$, where c is the number of codons, and f_c is the receptive field size per codon, $n_{CN} = f_c \frac{c}{R_S}$. K is a constant reflecting the relative

[8] In our experience and that of many others in the field, that stabilization time is more or less independent of the number CNs in the network, so network settling time is not a useful performance measure.

"conductor per unit area", and is a function of m_l and m_w. R_S is the intra-CN synapse redundancy, that is, the ratio of inputs to synapses (e.g., when R_S=1 each CN input is used once at the CN, when R_S=2 each input is used on the average at two sites of a single CN).

Only m_l levels of metal are possible, therefore if there is a large number of connections between widely distributed CNs, then the inter-CN area that is devoted to non-local interconnect grows rapidly.

Theorem 5: Assuming a rectangular ***unbounded*** grid of CNs (all CNs are equi-distant from their four nearest neighbors), where each CN has a square receptive field of CNs that is $2D$ squares on a side, the number of "wires" or connections, W, passing through a single square between four CNs (assuming an unbounded grid of CNs in all directions) is

$$W \leq 4[(D+1)^2-D] + \sum_{d=1}^{D-1}(8d+4)\left[(D+1)^2 - \frac{(D+1)^2 d}{d+1} - \frac{\sqrt{2}}{2}(d+\frac{1}{2})\right] \qquad (38)$$

Proof: The above relationship can be determined geometrically. Imagine a single square of the grid, each CN at the corner has an arc that subtends 90 degrees of a square that is the CN's receptive field. The area of that square is $(D+1)^2$. The total number of connections due to the four CNs surrounding the square is four times that, minus one row of D CNs along the edge (needed due to the discrete nature of the calculation and the desire to not count CNs along the edges twice), which gives $4[(D+1)^2-D]$.

Proceeding outward from the square one perimeter of CNs at a time, there are 12 CNs in the second perimeter, 20 in the next, etc. or $(8d+4)$ for each, where d is the distance from the center square, $1 \leq d \leq D-3$. The number of connections that pass through the square to each of these CNs is directly proportional to the area of square, since CNs are distributed evenly in this area, subtended by an angle between the target CN and the end of receptive field, as is shown in Figure 7. This area is equal to the area of this quadrant of the receptive field, $(D+1)^2$ minus the triangle on each side (B and C) and the triangle (D) of

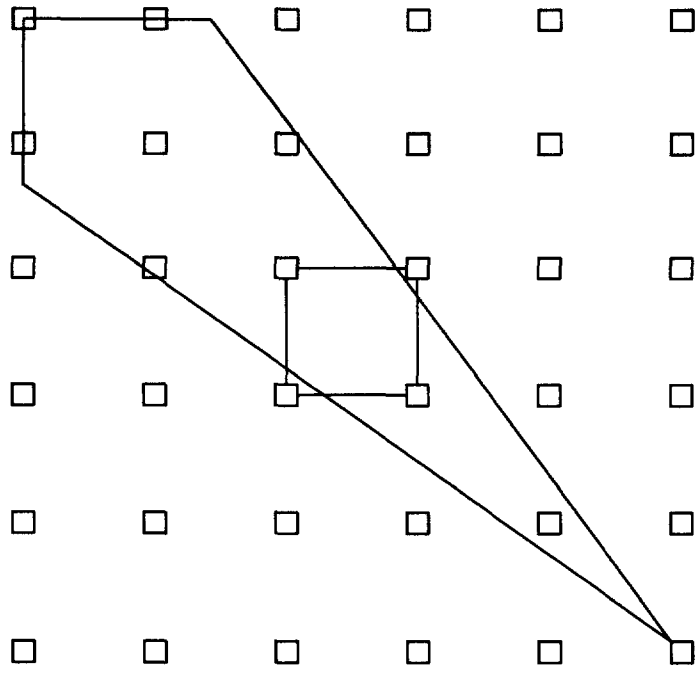

Figure 7 - Interconnection Calculation

area between the target CN and the square. The area of B and C is $\frac{(base)(height)}{2}$ where the height is $(D+1)$, and the base equals $\frac{(D+1)d}{d+1}$. The angle at the CN is the $\cos^{-1}$ of $\frac{d}{d+1}$. Since the angle is the same, the base is to $(D+1)$ as d is to $d+1$. Finally, the small triangle has base equal to $\sqrt{2}$ and the height is $\sqrt{2}(d+\frac{1}{2})$ - this is worst case and is for a triangle from a corner CN on the perimeter, CNs along the side will have somewhat less, so the bound is not exact (within a factor of $\sqrt{2}$). □

Corollary 6: With a square receptive field of $2D$ CNs on a side, the number of connections, W, that pass through a single square (with one CN at each corner) is

$$W=O(D^3); \tag{39}$$

Proof: By ignoring lower order terms and assuming $D \gg 1$, we get for the right hand side

$$D^2 \sum_{d=1}^{D} \left(d - \frac{d^2}{d+1} \right) = D^2 \sum_{d=1}^{D} \left(\frac{d^2+d-d^2}{d+1} \right) \tag{40}$$

which is approximately equal to D^3. □

To find the area of a square, it is assumed that, since there are approximately W signals passing through the square (entering and leaving), there are $\frac{W}{2}$ signals leaving each side. The metal area of the square then is

$$A_{MI} \equiv \left[\left(\frac{W}{2} \right) \left(\frac{m_w}{m_l} \right) \right]^2 \tag{41}$$

Let the receptive field size (in total connections to the CN) be n_{CN},

$$n_{CN} = (2D+1)^2 = O(D^2) \tag{42}$$

or

$$D = O(n_{CN}^{\frac{1}{2}}) \tag{43}$$

The total area, A, then is

$$A = O(n^3) \tag{44}$$

The total metal interconnect area increases as the *cube* of the per CN fan-in, and this is for a network with maximum locality!

Another option is to place all CNs along a diagonal, which gives n^2 area. However, this technique only works for a *bounded* number of CNs. Also, it is a different computational model, where dendritic computation is spread over a very large area which does not match all network models. The theorem proven here covers an infinite plane of CNs each with a *bounded* receptive field with more localized computation. Hybrid techniques may be able to break the n^3 to some degree, but that is highly algorithm/implementation dependent.

There is one major simplifying assumption here, and that is, that each CN's receptive field contains the *nearest* CNs on the plane. This is obviously not true in biological systems and will not be true in their silicon counterparts. However, since I am using *random* vectors, with independently, identically distributed elements, the "where" of the connections is not as important as their number. In future research more highly structured input will be used and there the structure of the connections becomes important. A broader, more diffuse interconnect (while maintaining a constant receptive field size) would increase metal area and performance. Likewise, such a model is closer to real systems. These types of connectivity patterns have not been characterized adequately here. For this paper it is assumed that only a few of a CN's connections come from outside the closest CNs, and that this number is small, so that these non-local connections do not greatly change the computed values of metal interconnect area.

4. SINGLE CN SIMULATION

Table I lists the analytic results for a single CN. f_v is the receptive field size of the CN, c is the number of codons, R_S is the input redundancy (the ratio of inputs to connections, $R_S \geq 1$), f_c is the order of each codon, A is the total area[9] cost in units of m_l/m_a, and m is that number of learned vectors where $Pcn=\epsilon$, that is, the maximum number of learned vectors (maximum capacity) where the information gain goes to zero. For this example, $\epsilon=0.1$. The general character of the results did not change significantly as ϵ was increased to 0.5 and these are not shown here.

[9] A is derived from a simplified version of equation 38.

As can be seen by the table, increasing the CN's receptive field size greatly improves the performance (allows larger m), but there is also an increasing cost, which increases faster than the performance! Another observation is that redundancy is quite effective as a means for increasing the effectiveness of a CN with constrained connectivity. Though not modelled here, there are some limits to R_S, since it can reach a point where the intra-CN connectivity approaches that of inter-CN for some situations. An order of magnitude increase in cost-effectiveness (A/m) is possible by increasing both order and redundancy.

A surprising effect is that, as expected, increasing the order of a CN increases the CN's performance (capacity), but not from f_c=1 to 2, where performance actually decreases. What happens is that the number of codons is decreased by 1/2 and the error introduced is much greater than that obtained by increasing the order of the codons from f_c=1 to f_c=2. As the order increases past 2, the error correcting capability of large codons increases faster than the error due to fewer codons. Because of the random, unstructured

f_v	R_S	f_c	c	m	A	A/m
256	1	4.00	64.00	128.00	3.334e+06	2.605e+04
256	1	2.00	128.00	104.00	3.242e+06	3.117e+04
256	1	1.00	256.00	128.00	3.227e+06	2.521e+04
256	2	4.00	128.00	256.00	3.457e+06	1.350e+04
256	2	2.00	256.00	224.00	3.273e+06	1.461e+04
256	2	1.00	512.00	272.00	3.242e+06	1.192e+04
256	3	4.00	192.00	416.00	3.580e+06	8.606e+03
256	3	2.00	384.00	320.00	3.303e+06	1.032e+04
256	3	1.00	768.00	416.00	3.257e+06	7.830e+03
4092	1	4.00	1024.00	2304.00	6.059e+09	2.630e+06
4092	1	2.00	2048.00	1792.00	6.057e+09	3.380e+06
4092	1	1.00	4096.00	2304.00	6.057e+09	2.629e+06
4092	2	4.00	2048.00	4608.00	6.061e+09	1.315e+06
4092	2	2.00	4096.00	3584.00	6.058e+09	1.690e+06
4092	2	1.00	8192.00	4608.00	6.057e+09	1.314e+06
4092	3	4.00	3072.00	7168.00	6.062e+09	8.458e+05
4092	3	2.00	6144.00	5632.00	6.058e+09	1.076e+06
4092	3	1.00	12288.00	6656.00	6.057e+09	9.101e+05

Table I - Analytic Bounds for a Single CN

nature of the input data space, one would expect that there is little higher-order information to take advantage of. Systems with highly structured input data should show significant improvement with higher-order codons (even f_c=2).

In order to verify the derived bounds, I also wrote a discrete event simulation of a CN, which used a random set of learned vectors, and where the CN's codons were programmed according to the model presented in section 2.0. Learned vectors were chosen randomly and subjected to random noise, ϵ. The CN then attempted to categorize each input into two major groups (CN output = 1 and CN output = 0). The categorization error is the output noise. Table II presents the results of a comparison between the analytic bounds and the simulation. R_S=1 for all simulations. *SimPcn* is the simulated CN output error, and *EstPcn* is the estimated or analytic CN output error (using the bounds derived in section 3.0).

For the most part the analytic bounds agree with the simulation, though they tend to be optimistic in underestimating the error. These differences can be easily explained by the simplifying assumptions that were made to make the bounds mathematically tractable. The next question then is what is the effect of connectivity variation when grouping CNs

f_v	c	ϵ	f_c	m	SimPcn	EstPcn
32	32	0.00	1	16	0.083123	0.018000
32	32	0.10	1	16	0.112889	0.128000
32	32	0.40	1	16	0.236964	0.384000
64	32	0.00	2	32	0.083123	0.064000
64	32	0.10	2	32	0.144383	0.106000
64	32	0.40	2	32	0.368711	0.374000
64	64	0.00	1	16	0.016711	0.016000
64	64	0.10	1	16	0.030368	0.048000
64	64	0.40	1	16	0.122627	0.376000
128	64	0.00	2	32	0.016711	0.006000
128	64	0.10	2	32	0.049372	0.034000
128	64	0.40	2	32		

Table II - Comparison of Analytic and Simulated CN, R=1

into larger systems? A detailed study of larger systems is beyond the scope of this paper, but a simulation of a simple recursive, auto-associative network was done.

5. MULTIPLE CN SIMULATION

Assume a single layer of CNs that are connected recursively such that the outputs for all CNs input to each CN. The result is a simple finite-state-machine. If the network exhibits positive information gain, then it will eventually stabilize. Such a network performs auto-association in a manner similar to Hopfield.

For bit vectors, auto-association is where a network stabilizes to the closest learned vector in terms of Hamming distance. Because there are asymmetrical connections, standard network stability proofs based on energy models ([Fel85]) do not apply. But, it can be shown that if there is always information gain, then a recursively connected system using random CN update always stabilizes.

The simulator for this model has several stages. In the first stage a connectivity graph is built. Here the user specifies the CN receptive field size (I tried both stochastic and deterministic connection selection with little difference in results). Next a set of learned vectors is created (these vectors are totally random, that is, no higher order structure is added). The connectivity graph and the learned vectors are used to program the codons (according to the model of section 2.0). This entire structure is then read into the simulator. The simulation is operated like a finite-state machine in that the CNs are initialized to some vector (a learned vector with added noise). The system runs sequentially, updating all CNs each clock, until it stabilizes, the output of the CNs of the previous clock being used as CN input in the next clock.

Synchronous update is more efficient than asynchronous update. Unfortunately, systems with synchronous update often fall into limit cycles. But this happened so rarely that synchronous update was used exclusively. One update of all CNs (to obtain a new set of outputs) is called a *cycle*. Settling time was almost always 1-3 cycles, in fact, in the hundreds of simulation runs, independent of network size, there were less than ten runs that required more than three cycles. This performance forcefully illustrates the $O(1)$ search behavior of parallel distributed models.

Table III shows the results for two small networks. Both networks have 64 CNs. The activation level indicates the expected number of 1's in the learned vectors - a lower percentage indicates a more localized representation. The *radius* is one half of the number of CNs in a CN's receptive field; f_c is the order of the codons; *select* is the expected number of codons, out of all possible codons, that were used (for f_c=1 all possible codons are taken, but for f_c>1, due to the increased number of possibilities, only 0.3 of all codons were used (these were selected randomly). *error* is the output noise, an input noise of 0.1 was used. There were 8 randomly chosen vectors in the training set.

The results show the affects discussed earlier, that is, by increasing the order, performance can be improved (since the information gain is increased). And, by increasing order and decreasing receptive field size, we can maintain performance for reduced connectivity, allowing us to trade-off non-local communication for local computation. Also, the higher order network is less sensitive to compression, which is shown by the fact that

Activation	n	radius	select	error
50%	1	32	1.0	0.32
12.5%	1	32	1.0	0.15
50%	2	10	0.3	0.09
12.5%	2	10	0.3	0.08

Table III - Recursive Network Simulation, 64 CNs

increasing the degree of distribution (by increasing the activation level) has less of an effect. Note that if we apply our cube law to the receptive field size, the total connectivity costs for the second and third networks were reduced by about 3^3=27.

The network performance is quite poor (except in the most favorable case, it adds noise to the input). One reason is that this network (the largest I could simulate easily on our VAX) was too small for our bounds to hold (there were too few codons). But the main reason was that the receptive field size was smaller than the input vector length. Since the input vectors were randomly selected, there was little higher-order information that could be used by the codons to decode correctly. It takes little compression to essentially disconnect two CNs whose receptive fields do not overlap. As a result, the collective network performance in the face of limited interconnect is even poorer than the single CN models predict.

6. DISTRIBUTED VS. LOCALIZED

Throughout this paper, it has been tacitly assumed that representations are distributed across a number of CNs, and that any single CN participates in a number of representations. In a *local* representation each CN represents a single concept or feature. It is the distribution of representation that makes the CN's decode job difficult, since distribution is the cause of the code compression problem.

There has been much debate in the connectionist/neuromodelling community as to the advantages and disadvantages of each approach; the interested reader is referred to Hinton [Hin84] and Baum et al. [BMW86]. Some of the results derived here are relevant to this debate. As the distribution of representation increases, the compression per CN increases accordingly. It was shown above that the mean error in a codon's response quickly

approaches 0.5, independent of the input noise. This result also holds at the CN level. For each individual CN, this error could be offset by adding more CNs. But adding more CNs is expensive and tends to eliminate one of the arguments in favor of distributed representations, that is, the multi-use advantage, where fewer CNs are needed because of more complex, redundant encodings. As the degree of distribution increases the required connectivity and the code compression increases, so the information content of the individual weights goes to zero (as they all approach 0.5).

My hope is that with the tools developed here, I will be able to show that the best representations for hierarchical/modular organizations will use representations that are partially distributed and partially localized.

7. SUMMARY AND CONCLUSIONS

In this paper a single CN (node) performance model was developed that was based on Communication Theory. Likewise, an implementation cost model was derived.

The communication model introduced the codon as a higher-order decoding element and showed that for small codons (much less than total CN fan-in, or convergence) code compression, or vector aliasing, within the codon's receptive field is a severe problem for large networks. As code compression increases, the information added by any individual codon to the CN's decoding task rapidly approaches zero.

The cost model showed that for 2-dimensional silicon, the area required for inter-node metal connectivity grows as the cube of a CN's fan-in.

The combination of these two trends indicates that past a certain point, which is highly dependent on the probability structure of the learned vector space, increasing the

fan-in of a CN (as is done, for example, when the distribution of representation is increased) yields diminishing returns in terms of total cost-performance. Though the rate of diminishing returns can be decreased by the use of redundant, higher-order connections.

One aspect of this paper (which can be viewed as fortunate or unfortunate depending on your point of view) is that it raises more questions than it answers. The treatment of multiple CNs with limited receptive field size operating in a larger network was for the most part missing, since I feel that it is important to understand the single CN first, before we move to larger networks.

The next step is to apply these techniques to ensembles of nodes (CNs) operating in a competitive learning or feature extraction environment.

8. REFERENCES

[ArS72] Arbib, M. A. and Szentagothai, J., "Conceptual Models of Neural Organization," *Neurosciences Res. Prog. Bull.*, vol. 12, 3 (October 1972), pp. 313-510.

[BaH86] Bailey, J. and Hammerstrom, D., "How to Make a Billion Connections," Tech. Report CS/E-86-007, Dept. of Computer Science/Engineering, Oregon Graduate Center, Beaverton, Oregon, July 1986.

[BHM] Baker, T., Hammerstrom, D. and McCartor, H., Adaptations of the Error Propagation Learning Algorithm for VLSI Implementation , Submitted to IEEE Conference on Neural Network Information Processing.

[Bal87] Ballard, D., "Modular Learning in Neural Networks," *Proc. of the AAAI Conf.*, Seattle, WA, August 1987, pp. 279-284.

[Bar85] Barlow, H. B., "Cerebral cortex as model builder," in *Models of the Visual Cortex*, D. Rose and V. G. Dobson (ed.), Wiley Interscience, New York, 1985, pp. 37-46.

[BSA83] Barto, A. G., Sutton, R. S. and Anderson, C. W., "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems," *IEEE Transactions of Systems, Man, And Cybernetics*, vol. SMC-13, 5 (September/October 1983), pp. 835-846.

[Bar85] Barto, A. G., "Learning by Statistical Cooperation of Self-Interested Neuron-Like Computing Elements," COINS Technical Report 85-11, Dept. of Computer and Information Science, University of Massachusetts, April 1985.

[BMW86] Baum, E. B., Moody, J. and Wilczek, F., "Internal Representations for Associative

Memory," Technical Report NSF-ITP-86-138, Institute for Theoretical Physics, Santa Barbara, CA, 1986.

[CaG86] Carpenter, G. and Grossberg, S., "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine," in *Computer Vision, Graphics, and Image Processing*, 1986. In press.

[CoG83] Cohen, M. A. and Grossberg, S., "Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-13, 5 (Sept/Oct 1983), pp. 815-826.

[CoH67] Cover, T. M. and Hart, P. E., "Nearest Neighbor Pattern Classification," *IEEE Transactions on Information Theory*, vol. IT-13, 1 (January, 1967), pp. 21-27.

[DSM85] Desimone, R., Schein, S. J., Moran, J. and Ungerleider, L. G., "Contour, Color and Shape Analysis Beyond the Striate Cortex," *Vision Res.*, vol. 25, 3 (1985), pp. 441-452, Pergamon Press Ltd..

[FeB82] Feldman, J. A. and Ballard, D. H., "Connectionist Models and Their Properties," *Cognitive Science*, vol. 6(1982), pp. 205-254.

[Fel85] Feldman, J. A., "Energy and the Behavior of Connectionist Models," Tech. Rep. 155, Dept. of Computer Science, Univ. of Rochester, Rochester, NY, November 1985.

[FMI83] Fukushima, K., Miyake, S. and Ito, T., "Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-13, 5 (Sept/Oct 1983), pp. 826-834.

[Fuk84] Fukushima, K., "A Hierarchical Neural Network Model for Associative Memory," *Biological Cybernetics*, vol. 50(1984), pp. 105-113.

[Gal68] Gallager, R. G., *Information Theory and Reliable Communication*, John Wiley and Sons, New York, 1968.

[Gol86] Golden, R. M., "A Developmental Neural Model of Visual Word Perception," *Cognitive Science 10*, 1986, pp. 241-276.

[Gro80] Grossberg, S., "How Does a Brain Build a Cognitive Code?," *Psychological Review*, vol. 87, 1 (January 1980), pp. 1-51.

[Gro82] Grossberg, S., "The Temporal Unfolding and Stability of STM and LTM Patterns," in *Competition and Cooperation in Neural Nets*, S. Amari and M. Arbib (ed.), Springer-Verlag, Berlin, 1982, pp. 295-341.

[Gro86] Grossberg, S., *The Adaptive Brain (I and II)*, Elsevier/North-Holland, Amsterdam, 1986.

[Ham86] Hammerstrom, D., "A Connectionist/Neural Network Bibliography," Tech. Report CS/E-86-010, Dept. of Computer Science/Engineering, Oregon Graduate Center, Beaverton, Oregon, August 1986.

[HMT86] Hammerstrom, D., Maier, D. and Thakkar, S., "The Cognitive Architecture Project," *Computer Architecture News*, vol. 14, 1 (January 1986), pp. 9-21, ACM SigArch.

[Hin84] Hinton, G. E., "Distributed Representations," Tech. Rep. CMU-CS-84-157, Computer Science Dept., Carnegie-Mellon University, Pittsburgh, PA 15213, 1984.

[HSA84] Hinton, G. E., Sejnowski, T. J. and Ackley, D. H., "Boltzmann Machines: Constraint Satisfaction Networks that Learn," Technical Report CMU-CS-84-119, Computer Science Dept., Carnegie-Mellon University, Pittsburgh, PA 15213, May

1984.

[Hop82] Hopfield, J. J., "Neural networks and physical systems with emergant collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79(April 1982), pp. 2554-2558.

[Koh84] Kohonen, T., *Self-Organization and Associative Memory*, Springer-Verlag, Berlin, 1984.

[Lin86a] Linsker, R., "From Basic Network Principles to Neural Architecture: Emergence of Orientation Columns," *Proc. Natl. Acad. Sci. USA*, vol. 83(November 1986), pp. 8779-8783.

[Lin86b] Linsker, R., "From Basic Network Principles to Neural Architecture: Emergence of Spatial-opponent Cells," *Proc. Natl. Acad. Sci. USA*, vol. 83(October 1986), pp. 7508-7512.

[Lin86c] Linsker, R., "From Basic Network Principles to Neural Architecture: Emergence of Orientation-selective Cells," *Proc. Natl. Acad. Sci. USA*, vol. 83(November 1986), pp. 8309-8394.

[Lyn86] Lynch, G., *Synapses, Circuits, and the Beginnings of Memory*, MIT Press, Cambridge, MA, 1986.

[Mar70] Marr, D., "A Theory for Cerebral Neocortex," *Proc. Roy. Soc. London*, vol. 176(1970), pp. 161-234.

[MGL86a] Maxwell, T., Giles, C. L., Lee, Y. C. and Chen, H. H., "Transformation Invariance Using High Order Correlations in Neural Net Architectures," *Proceedings International Conf. on Systems, Man, and Cybernetics*, 1986.

[MGL86b] Maxwell, T., Giles, C. L., Lee, Y. C. and Chen, H. H., "Nonlinear Dynamics of Artificial Neural Systems," in *Neural Networks for Computing*, American Institute of Physics, 1986.

[McC65] McCulloch, W. S., *Embodiments of Mind*, MIT Press, Cambridge, MA, 1965.

[McK85] McKay, D. M., "The significance of 'feature sensitivity'," in *Models of the Visual Cortex*, D. Rose and V. G. Dobson (ed.), Wiley Interscience, New York, 1985, pp. 47-53.

[MiP69] Minsky, M. and Papert, S., *Perceptrons*, The MIT Press, Cambridge, MA, 1969.

[Mou77] Mountcastle, V. B., "An Organizing Principle for Cerebral Function: The Unit Module and the Distributed System," in *The Mindful Brain*, MIT Press, Cambridge, MA, 1977.

[Par85] Parker, D. B., "Learning-Logic," Technical Report Tech. Rep.-47, Center for Computational Research in Economics and Management Science, MIT, Cambridge, MA, April 1985.

[RCE82] Reilly, D. L., Cooper, L. N. and Elbaum, C., "A Neural Model for Category Learning," *Biol. Cybern.*, vol. 45(1982), pp. 35-41.

[RoD85] in *Models of the Visual Cortex*, D. Rose and V. G. Dobson (ed.), Wiley Interscience, New York, 1985.

[RHW85] Rumelhart, D. E., Hinton, G. E. and Williams, R. J., "Learning Internal Representations by Error Propagation," ICS Report 8506, Institute for Cognitive Science, La Jolla, CA, September 1985.

[RuM86] D. E. Rumelhart and J. L. McClelland, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1 and 2, Bradford Books/MIT Press, Cambridge, MA, 1986.

[ShW49] Shannon, C. E. and Weaver, W., *Mathematical Theory of Communication*, The University of Illinois Press, 1949.

[ShF85] Shastri, L. and Feldman, J., "Evidential Reasoning in Semantic Networks: A Formal Theory," *Proc. 9th IJCAI*, 1985, pp. 467-474.

[Smo86] Smolensky, P., *Information Processing in Dynamical Systems: Foundations of Harmony Theory*, vol. 1, Bradford Books/MIT Press, Cambridge, MA, 1986. pp. 194-281

[SoP87] Somani, A. K. and Penla, N., "Compact Neural Networks," *Proc. of the First International Conference on Neural Networks*, San Diego, CA, June 1987.

[Wic79] Wickelgren, W. A., "Chunking and Consolidation: A Theoretical Synthesis of Semantic Networks, Configuring in Conditioning, S-R Versus Cognitive Learning, Normal Forgetting, the Amnesic Syndrome, and the Hippocampal Arousal System," *Psychological Review*, vol. 86, 1 (1979), pp. 44-60.