

# Integration and Hybridization in Neural Network Modelling

Wesley Royce Elsberry

Presented to the Faculty of the Graduate School of  
The University of Texas at Arlington in Partial Fulfillment  
of the Requirements for the Degree of

Master of Science in Computer Science

The University of Texas at Arlington  
August 1989

# Acknowledgements

I wish to thank the many people who have made my graduate program a rewarding, enlightening, and interesting experience. My especial thanks to Bob Weems, Ken Youngers, Vijay Raj, Steve Hufnagel, and Farhad Kamangar for their exemplary instruction. The advice and encouragement of Bill Buckles was critically important to this refugee from the life sciences. The members of my graduate committee, Karan Briggs, Lynn Peterson, and Daniel Levine, have provided me with technical resources, instruction, referrals, and general good advice in plenty. Sam Leven provided the example problem description, the classical sequence data, and much of his own time and expertise to aid me in developing both the simulation program and my understanding of the field. I am indebted to Dr. Levine for his great interest in teaching the principles of cognitive modelling to as wide an audience as possible, for without his express encouragement I would not have become acquainted with the field, and also for his considerable personal assistance in developing this thesis. The enthusiasm of Harold Szu helped to motivate me to undertake a deeper inquiry into neural network modelling. Finally, without the continual support of my spouse, Diane Blackwood, this thesis and the classroom work which formed the basis for it would not have been possible.

July 28, 1989

# Abstract

Artificial neural network models derived from different biological behaviors or functions can be used in an integrative fashion to create an extensive problem-solving environment. An example problem of limited melodic composition is approached by the use of Hopfield-Tank, back-propagation, and Adaptive Resonance Theory networks serving as plausible next note generator, musical sequence critic, and novelty detector, respectively. Biological bases for integrative function are discussed, and the experimental role of synthetic systems such as the example integrated network is explored.

# Contents

1	Defining The Role And Nature Of Artificial Neural Network Modelling	5
2	Integration In Neural Network Modelling	20
3	Example Problem	25
4	Results	30
5	Discussion	32
A	Sample Melody Output Of The Various Note Generator Programs	37
B	Program Source Listing: Integrated Ann Note Generator	38
C	Program Source Listing: Back-Propagation Unit	39
D	Program Source Listing: List Structures Unit	40
E	Program Source Listing: Miscellaneous Procedures Unit	41
F	Program Source Listing: Global Type And Variables Unit	42
G	Program Source Listing: Classical Instructor Unit	43
H	Program Source Listing: Ansi Screen Control Unit Unit Ansi_Z;	44
I	Program Source Listing: Musical Sequence Evaluator Program	45
J	Program Source Listing: Random Note Generation Program	46
K	Program Source Listing: Note Sequence Playing Program	47
L	Program Source Listing: Rule-Based Note Sequence Generation Program	48
M	Program Source Listing: Offline Back-Propagation Network Training Program	49
N	Data File Listing: Hopfield-Tank Network Weight Data File	50

O	Data File Listing: Classical Sequences Data File	51
P	Data File Listing: Back-Propagation Network Data File	52
Q	Program Source Listing: Translator From Program Note Files To Music Transcription System Song Format	53

# Chapter 1

## Defining The Role And Nature Of Artificial Neural Network Modelling

### Cultural Bias and Its Application to Cognitive Inquiry

Plato believed that there existed a pure world of ideas, whose perfect forms were poorly mimicked by copies in the reality apparent to our senses. This concept of the dominant, lofty nature of ideas and the thoughts that manipulated those ideas would do more to inhibit inquiry into material processes than virtually any other single cause in the succeeding two millennia. The legacy of this philosophical outlook still permeates our culture, coloring the basic assumptions and viewpoints of researchers even after extended scientific training.

The firmness of general belief in the separate existence of ideas contributed to the Lamarckian hypothesis of the inheritance of acquired characters. While there were critics of Jean Baptiste Lamarck's work from its introduction, these were by no means an overwhelming group by numbers. The rediscovery of Gregor Mendel's work in 1905 provided clear and convincing evidence against Lamarck's hypothesis, yet even Watson and Crick's elucidation and characterization of DNA failed to dispel the last vestiges of belief in the inheritance of acquired characters. The ready and continued acceptance of the Lamarckian hypothesis into the mid-twentieth century gives an indication of the continued influence of Plato's world of ideas, even when confronted with contradicting evidence and convincing counterarguments.

Biology, however, was by no means the only scientific discipline to be touched by the cultural bias of Platonic ideals. Physics experienced great upheaval and internal dissent as the deterministic view of Newton and Laplace gave way to the quirkiness and Heisenbergian uncertainty of quantum mechanics (Hawking 1988).

In thinking about thinking, researchers have tended to denigrate approaches dependent upon investigation of physical processes. The high regard given ideas and thought processes has nearly always provided Western observers with a sense of revulsion at even considering that such

sacred items could be solely the product of soft, squishy brain tissues and their component parts, neurons. This could be considered somewhat akin to Roquentin's nausea upon consideration of radical contingency (Ferguson 1987).

Our culture does not encourage modes of inquiry that tend to displace the traditional view of the mind as the only known organ with which the perfect world of ideas can be perceived. It especially does not encourage the denial of the Platonic ideal world, yet it has been quite some time since any reputable scientist has explicitly advanced support for that concept.

Cognitive science, however, must deal explicitly and forthrightly with the subject of ideas and thoughts. In a field where cultural bias has, can, and will directly effect the discussion of the subject at hand, it pays to recognize the existence and probable magnitude of that bias. I have briefly delineated the persistence and magnitude of our Platonic bias in biology and physics; I will assert that the bias is greater in cognitive science, where it may be more directly challenged.

#### Artificial Intelligence: Definitions

Artificial intelligence describes at once a sub-discipline of cognitive science and the goal of that sub-discipline. The goal is the description and production of an artificial system or systems that can be described as intelligent in operation, function, or effect, in both global and local contexts. As with most complex fields, there are several ways in which to approach direct inquiry into the topic. In what has been termed the "top-down modelling school" of artificial intelligence, the emphasis has been upon systems of formal logic and other explicit symbol manipulation techniques. As may have been apparent from the previous description, there is a "bottom-up modelling school" of artificial intelligence. This school of inquiry seeks to examine the basic processes that are known to produce intelligent action in humans, those basic processes of neural function and coordination.

By extension, the bottom-up modelling school is concerned with neural function in general. This is considered necessary because of the great complexity of biological neural systems. While much research has been done and continues to be done, there exists no basic understanding of the detailed structure and operation of biological neural systems. There is a wealth of data, but a paucity of organizing principles. The bottom-up modelling school attempts to provide possible organizing principles, testing these through a process of modelling and incremental design improvements. This area of research has become known as artificial neural network modelling.

The definitions given above differ somewhat from what may be considered standard in the artificial intelligence community. In their Turing Award speech, Newell and Simon state:

The notion of physical symbol system had taken essentially its present form by the middle of the 1950's, and one can date from that time the growth of artificial intelligence as a coherent subfield of computer science. The twenty years of work since then has seen a continuous accumulation of empirical evidence of two main varieties. The first addresses itself to the sufficiency of physical symbol systems for producing intelligence, attempting to construct and test specific systems that have such a capability. The second kind of evidence addresses itself to the necessity of having a physical symbol system wherever intelligence is exhibited. It starts with Man, the intelligent system best known to us, and attempts to discover whether his cognitive activity can be explained as the working of a physical symbol system. . . . The first is generally called artificial intelligence, the second, research in cognitive psychology. (Newell and Simon 1976)

This formulation of definitions seems at once too narrow and not properly descriptive. It is too narrow in that it limits severely the range of activities which may be considered artificial intelligence research.

While Newell and Simon talk of physical symbol systems, there is the strong implication that they refer only to computational methods of explicit symbol manipulation, as in the use of languages such as LISP and Prolog. The definition of the second part is too narrow, in that it postulates no overlap between the fields of artificial intelligence and cognitive psychology, and also provides no linkage for concepts of operation derived from empirical studies of biological intelligent systems to be incorporated into an artificial intelligence framework. As stated later by Newell and Simon,

The symbol system hypothesis implies that the symbolic behavior of man arises because he has the characteristics of a physical symbol system. Hence, the results of efforts to model human behavior

with symbol systems become an important part of the evidence for the hypothesis, and research in artificial intelligence goes on in close collaboration with research in information processing psychology, as it is usually called.

This seems to indicate that the previously given definitions were not truly descriptive of the relationship between artificial intelligence and other components of cognitive science. By including artificial intelligence as a sub-discipline of cognitive science, it becomes clear that artificial intelligence is not disjoint from research into considerations of what constitutes intelligence.

An alternative formulation for defining artificial intelligence in more general terms is given by Charniak and McDermott (1985), where they state that artificial intelligence "is the study of mental faculties through the use of computational models." This form of definition allows for the top-down and bottom-up approaches to artificial intelligence to be given co-equal rank as complementary research disciplines.

Artificial Neural Network Modelling

Artificial neural network modelling, the bottom-up school of artificial intelligence, derives from the use of biological nervous systems as suitable exemplars for an approach to coordination, cognition,



and control in artificial systems. This approach, in one form or another, has been with us for a long time.

In the 1940's, McCulloch and Pitts demonstrated the possibility of casting boolean logic systems into networks of thresholded logic units. This certainly fits the Newell-Simon definition of a physical symbol system. McCulloch and Pitts' seminal paper and subsequent work introduced a new concept for consideration: rather than driving the understanding of biological cognitive processes through increasing sophistication of technology, perhaps technology can be made more capable and sophisticated by elucidating mechanisms of function from biological processes involved in cognition (Levine 1983, 1990).

This differs qualitatively from the viewpoint common to many observers of human intelligence. There has been a tendency for people to compare the operation of thought processes with the current leading edge of technology. Freud compared the mind to a steam engine, Twain would speak of "mill-works" in relation to speech production, and various persons in this century have blithely and confidently settled upon the computer as a fully analogous system. A circular system can be noted here, though, as the image of the mind as being like some mechanism helped contribute to such endeavors as Pascal's calculator, Babbage's Analytical Engine, and the Hollerith punched card controlled census tabulator. These efforts, in turn, inspired the modern computer. (The influence of the two World Wars upon the motivation for the development of the computer must also be acknowledged.) However, as the Metropex Study Group on Computational Neuroscience (1988) points out,

Computational study of brain structures began in the 1940's when digital computers were emerging. The first computer architectures were developed in a collaboration between the mathematicians John von Neuman and Norbert Wiener working closely with physiologists like Warren McCulloch. Early computer scientists were captivated by the analogy between the all-or-none action potentials of the neuron and binary switches representing bits in computers. Models of biological computation thus had a seminal role in the design of modern computer architecture.

Thus, while comparisons of the workings of the mind to technology helped to motivate research into further technological advances, the insight into the successful design for a significant new technology, computers, came instead from paying attention to the actual mechanisms of brain function. The computer has come to be pressed into service as a "new" metaphor for thought processes in biological systems. While the use of technological metaphors for cognition is persistent, having produced quaint turns of speech and aphorisms, it has not been significantly conducive to a direct understanding of the actual cognitive processes being described. The turnabout analogy produced by McCulloch and Pitts has provided far more benefits to cognitive science and computing through its establishment of neural modelling. Binary switching technology failed to provide significant insight into biological neural activity and

function. Leon Harmon (1970) noted that "[a]nother kind of difficulty in coming to an understanding of nervous systems is that we may be conditioned into thinking about them in ways that are more constrained than we like to admit or are sufficiently aware of." The cultural identity which we share through environment can be a handicap in research into the nature and mechanisms of our intelligence.

The model networks of McCulloch and Pitts, an example of which is shown in Figure 1, were seen to provide useful mechanisms and insights for computing machinery. These networks consist of threshold logic units receiving excitatory and inhibitory inputs of integer value. A simple sum of these values is compared with a threshold level for the unit, and the unit is considered to "fire" if its inputs equal or exceed the threshold value. A unit's activity is then carried forward to further connected units in the system. McCulloch and Pitts (1943) proved that any boolean function could be created by the appropriate combination of such units.

The McCulloch-Pitts formalism is quite similar to principles of design using TTL logic, as diagrams can attest. It should be noted that McCulloch and Pitts described their formalism in 1943, some sixteen years before the invention of the integrated circuit. While significant problems reduce the overall utility of McCulloch-Pitts networks, modified forms continue to be advanced (Szu 1989).

Given the basis of a neural computational framework, additional features extracted from biological research were gradually added to proposed models. The idea that inputs to neurons had graded values modified by the efficiency of synaptic connections led to several important advances. Hebb proposed a synaptic modification rule to allow a form of learning in randomly connected networks (Hebb 1949, Levine 1983). His formulation was that the efficiency of a synapse which connects two neurons increases if both of the neurons have high activities at the same time. This relatively simple rule has a number of drawbacks, which have since been extensively cataloged and discussed. However, it cannot be denied that Hebb's Law, as it has been called, was an important advance for modelling.

The idea of stating an organizational principle for network design also represented an important advance for artificial neural network modelling. Rosenblatt's Perceptron architectures demonstrate this well. Rosenblatt postulated several network models, among these was a three-layer network. The layer which received external inputs was the sensory layer, and the layer which gave an output (either as a raw signal or interpreted as a motor output) was termed the response layer. An associative layer of neurons provided the articulation between sensory and response layers (Levine 1983). This general method of organization helped move artificial neural network modelling away from specifically designed networks and randomly organized network models.

The drawbacks associated with the special design of network models remain a concern for ANN modellers today. Such systems are fragile or brittle, meaning that a fault in any part of the network could cause a general failure of function. Also, if an error in design occurred, the resulting network would have no method of overcoming the design fault. This particular pitfall is known by the assumption necessary for correct function to be achieved: "programmer omniscience." Since programmer omniscience cannot be guaranteed, systems predicated on this principle are inherently unreliable (Hecht-Nielsen 1986).

Designing with generality as an important principle of function results in artificial neural networks that are described as self-organizing. A self-organizing network will conform its function to the particular problem or context through a process of adaptation or learning.

#### Current Models Narrowly Focussed

The tendency in developing an artificial neural network model is to constrain its function to a well-defined domain. This design principle enables better control over evaluation of the functionality of the design. Data from neurological or behavioral studies dealing with the problem domain can then often be directly applied to either training or evaluating the model.

There exist many general functions that are commonly encountered in biological neural networks which can be considered to have important implications for computational study. Among these are included associative memory, classification, pattern recognition, and function mapping. ANN models proposed to implement these functions include Bi-directional Associative Memory, Adaptive Resonance Theory (classification), Brain-State-in-a-Box (classification), Neocognitron (invariant pattern recognition), and Back-Propagation (function mapping) (Simpson 1988).

Unfortunately, the underlying reason for existence of each model is overlooked in comparing different models. The human tendency to wish to attribute a single scalar quantity denoting how "good" an ANN model is will often cause people to overlook the fact that certain networks should not be compared as being equivalent. Lippmann provides a good basic overview of six different ANN models, yet falls prey to this pitfall. Each of the models is interpreted as a classification network, yet only the Hamming and ART networks are designed to function as classifiers (Lippmann 1987).

#### Multi-architecture Integrative Systems

The use of multiple complementary architectures in designing application systems has not often been explored. While this approach to system design is commonly touted in ANN simulation aid advertisements, it is less frequently featured in the literature. In an advertisement for the ANZA Neurocomputer, HNC Corporation states, "In these networks the interconnect geometry is already determined and the form of the transfer

equations is fixed. However, the number of processing elements (neurons), their initial state and weight values, learning rates, and time constants are all user selectable, thus allowing one to customize a particular network paradigm (or combination of paradigms) to suit a particular application." There are some reasons for the relative lack of multi-paradigm (multi-architecture) research results in publication.

The design principles stated earlier still hold, that it is easiest to design an architecture with a narrow definition of function. There is the tendency for such architectures to incorporate simplifying assumptions which aid in simulation or real-world implementation ("casting in silicon," as the phrase goes). The ease of verifying correct operation is increased for architectures which have a narrowly defined application, as relatively clean data or simple theoretical findings are likely to be available against which the architecture may be evaluated.

The relative complexity of modelling for even constrained contexts provides another reason for concentration upon single paradigm systems. The number of parameters which can effect a model's performance ranges from zero for linear systems, such as Widrow's ADALINE (Widrow 1987), to tens of scalar parameters, as may be encountered in Carpenter and Grossberg's Adaptive Resonance Theory architecture (Carpenter and Grossberg 1987a, 1987b; Simpson 1988). When dealing with nonlinear dynamical systems for which no closed-form solution exists, slight changes of system parameters can result in large scale changes in behavior. Carpenter and Grossberg point out relative constraints upon the parameters used in the ART 1 architecture, giving guidelines for values which should yield stable operation of the network. Finding suitable parameters for operation of a particular architecture can be a frustrating and time-intensive experience.

As an example, an architecture called the "on-center, off-surround" network (OCOS) is based upon a relatively simple equation that appears in slightly modified form in many competitive networks. One form of this equation (Grossberg 1973) is:

$$\begin{aligned} dx/dt = & -A X_i \text{ [first term]} \\ & + (B-X_i) (I_i+f(X_i)) \text{ [second term]} \\ & - (X_i+C) \left( \text{Sum from } k = 1 \text{ to } n \text{ of } ((k \neq i) (J_k+f(X_k))) \right) \text{ [third term]} \end{aligned}$$

(Eq. 1)

The first term represents decay of activity over time; the second term represents increase in activity due to excitatory input values,  $I_i$ , and recurrent self-excitation; the third term represents decrease in activity due to inhibitory input values,  $J_k$  and competition from other nodes in the network. For each term, there is an associated parameter. A hidden parameter of the architecture is a factor by which to multiply the result of the difference equation. This is usually set to be much less than one, especially when using the simple forward Euler method of system updating. However, there is a performance trade-off involved with setting

this factor too small: the system will converge to a stable result, but will take a long time to converge. If the factor is set to a large value the network will "blow-up," which is a state of wild fluctuation in activation values for the nodes in the network. Such a system cannot converge. The solution is to find a value for this factor which will ensure convergence without an inordinate amount of time taken in reaching convergence. The explicit parameters given in the equation must be matched to the size of the other parameters to maintain set relationships. In order to find parameter values, I have used a network simulation which runs through permutations of sets of discrete parameter values. By examining the resulting output which indicated whether the network run achieved convergence and the number of time steps necessary to achieve convergence, I was able to settle upon a suitable choice of parameters for regular use and was able to decipher a coarse set of relationships between the parameters.

In consideration of these complexities in modelling and simulation, synthesis of complex systems from simpler subunits has some significant obstacles to overcome. There are more degrees of freedom for system operation, leading to complexities in articulation and coordination of subunits. This may increase the necessary number of system parameters, increasing complexity in a combinatorial fashion. Usually, there will either be less data available for evaluation of the complete system behavior, or the data that is available will be uncertain. The general principles to apply to a synthetic endeavor in artificial neural network modelling remain to be elucidated.

In addition, synthesis has traditionally been given short shrift in Western culture (Paris 1989). The application of synthesis in philosophy has largely been confined to the works of Wittgenstein and Marx. Marx, of course, developed a philosophy whose application was inimical to most Western economic structures. This has caused both his work and his approach to be considered with contempt.

On the positive side, however, synthesis can lead to significant insights as different assumptions and features can be applied to functional design. The process of integration and hybridization in artificial neural network modelling may be expected to lead to new insights into structure and function of benefit to artificially intelligent systems, given the past history of benefits derived from consideration of concepts borrowed from disparate disciplines. In this case, the complexity of knowledge or topic of a contributing discipline can provide a measure of the possible beneficial overlap: as complexity increases, the set of concepts which may profitably be applied to the new context also increases. As this rule would predict, biological nervous systems provide a very large pool of concepts ready for reconstruction in the artificial intelligence framework.

Biological Neural System Complexity

The complexity which drives that expectation can be readily appreciated, as in the human nervous system the number of neurons is in the billions, and each neuron will have connection to between tens and tens of thousands of other neurons. This physio-spatial complexity pales when compared to the complexity that can result when one attempts to define a state for the neuron. The state can be considered to be a combination of electrical activity, ionic balance, hormonal levels, input activities, and many other factors. Since for several of these attributes there exist separate contributing factors, the number of components contributing to neuron state can easily exceed twenty, and there may be hundreds of such components. Add to this that these components often have analog values, separate time scales of action, and capacity for permanent change in the neuron's response, and the possible state function for a single neuron displays a suitably bewildering complexity. Since this level of complexity is not conducive to direct examination, the process of problem decomposition and solution is applied. The implication for artificial neural network modelling is that certain gross features are modelled. Typically, a model assumes that the major interesting component of neural function is the electrical activity of the contributing neurons or nodes. A further assumption in general currency is that the electrical activity of an ANN may be modeled with the elements or nodes responding in the manner expected of neural populations rather than individually thresholded neurons (Levine 1983). Some models take into account generalized neurotransmitter effects, with Grossberg's gated dipole model providing a good example (Grossberg 1972). These simplifying assumptions still leave a high level of complexity to be dealt with in developing ANN designs and applications.

#### Relevance of Biological Models

Biology provides the exemplars for self-organizing adaptive systems which have proven useful in artificial neural network modelling thus far. While strict adherence to biological accuracy in modelling may well be counter-productive to advances in modelling techniques, rejecting biological principles may then lead to difficulties in later integrative work.

The biological framework of neurophysiological function provides a basic structure which makes for a common ground of interaction in models. For example, the range of activation and output of neurons is rather strictly proscribed, while the range of activation of an analogous unit in neural network models is limited only by the capability of the processor in specifying a floating point exponent. This would make interfacing two models using different ranges of activation less straightforward.

The inherent complexity of biological systems can support two different arguments concerning the future of modelling. As Hawking (1988) says, describing a complex system all at once is terrifically difficult.

It is much simpler to model several different components of the overall

system. Bringing these partial models together to form a complete model can become problematic, as in Hawking's example of the search for the Grand Unified Theory in physics. In one sense, then, the effort to integrate models of limited function may be premature. However, it may be that along with the incremental advances made in modelling small subunits of biological neural function, we should also attempt incremental integration of well-understood low level models. This would help prevent the kind of situation which exists in physics, with two or more highly disjoint models prevailing and no unification or integration yet in sight. Since biological systems are inherently more complex than the physical systems upon which they are based, it becomes important to keep an eye on the eventual need for integration and resolution of subsidiary models. Models which utilize features from two or more extant ANN architectures such as Hecht-Nielsen's Counterpropagation Network (Hecht-Nielsen 1987) demonstrate the useful qualities which synthesis of models can bring to functionality. The counterpropagation network is derived from Kohonen's self-organizing map model and Grossberg's competitive learning networks. Hecht-Nielsen notes that this network is designed for function mapping and analyzes its performance as compared to the back-propagation architecture. While the general functionality of counterpropagation networks (CPN) remains lower than that of back-propagation networks, there exists a subclass of mapping functions for which the CPN will train faster, and there also exists a closed-form solution for the error of the CPN. As Hecht-Nielsen notes, "Finally, CPN illustrates a key point about neurocomputing system design. Namely, that many of the existing network paradigms can be viewed as building block components that can be assembled into new configurations that offer different information processing capabilities."

#### Artificial Neural Network Models: Three Architectures

For the purposes of exploring multi-architecture system design, I selected three artificial neural network architectures for incorporation into the overall system. These were the Hopfield-Tank network, back-propagation, and Adaptive Resonance Theory 1 models. These networks perform three different functions: Hopfield-Tank is used for optimization or constraint satisfaction, back-propagation is a general-purpose mapping network, and Adaptive Resonance Theory 1 is a classifier network.

#### Hopfield-Tank Networks

In 1982, John Hopfield wrote an article, "Neural networks and physical systems with emergent collective computational abilities," which was published in Proceedings of the National Academy of Sciences, describing a model of neural computation which was readily implementable in current solid-state technology. This article and the model which it described has been widely credited with a resurgence of interest in ANN modelling.

The network architecture presented by Hopfield and Tank (1985),

hereafter known as HTN for "Hopfield-Tank Network[s]", is a single-layer fully interconnected network model (Figure 2). There is no learning rule for this network, although various researchers have proposed modified HTN architectures that do incorporate adaptive learning. Weights between nodes in an HTN are fixed and symmetric, and connections between a node and itself are zero. The advantages of these criteria for network design are that the system dynamics can be shown to perform an "energy minimization" in reaching a stable state. When the weights are determined according to system constraints, the system can be characterized by a Liapunov function, providing a measure for system energy. HTN's have been applied to various constraint satisfaction and optimization problems. Hopfield and Tank attracted much attention by demonstrating the utility of an HTN in generating good solutions to the "Traveling Salesman Problem" (or TSP), a non-polynomial time complete problem. The TSP can be described as choosing a minimum length path among Figure 2

a set of cities such that each city is visited once, and the salesman returns to the city of origin. The closed path length constitutes the measure for any particular solution. As the number of cities increases, the number of possible valid tours increases combinatorially. However, it can be shown that an HTN will produce good tours in constant time. An HTN used for TSP computation does not necessarily converge upon the optimum solution, but will reject non-optimum solutions and give relatively "good" solutions. Unfortunately, it can also be shown that the HTN trades off constant time operation for  $O(n^2)$  space considerations.

Figure 2 shows the HTN architecture for solution of the TSP.

Although the nodes in the network are treated as elements in a vector, the visual representation which makes the most sense to the designer and other interested humans is a matrix of nodes. By imposing this structure upon our view of the network, we can associate each row with a city in the tour, and each column with the position of a city in the final tour (City  $n$  is in Position  $m$  in the tour). The constraints of importance are that for a valid tour, each city appears once, and each position in the tour has only one city. Thus, for the state of the network at equilibrium, we should find high activities in  $n$  nodes, where  $n$  is equal to the number of cities or positions. The interconnections between nodes in a row or column are inhibitory, causing highly active nodes to reduce the activity of other nodes in the same row or column. The interconnections from a node to its neighbors in adjacent columns are proportionally more inhibitory as distance between cities represented by those nodes increases.

Hopfield-Tank Equation

The equation defining the network's activity over time (Hopfield and Tank 1985) is

$$C_i (du_i/dt) = (\text{Sum from } j = 1 \text{ to } N \text{ of } (T_{ij}V_j - u_i/R_i + I_i)) \quad (\text{Eq. 2})$$



where  $i$  and  $j$  designate neurons in the network,  
 $N$  is the total number of neurons in the network,  
 $T$  is a connection weight between neurons,  
 $V$  is the output value function for a neuron,  
 $u$  is the activity of a neuron,  
 $I$  is the external input for a neuron,  
 $C$  is the capacitance of a neuron,  
and  $R$  is the resistance of a neuron.

#### Back-propagation Networks

Back-propagation is a short version of "correction by the backward propagation of errors." The learning rule used in back-propagation networks (BPNs) is termed the generalized delta rule (Simpson 1988; Rumelhart, Hinton, and Williams 1986). Basically, a BPN is a multi-layer (with at least three layers) network whose nodes use a sigmoid output function. The BPN will map a set of input activities to another set of output activities, given training upon a set of example input/expected output vector pairs. The BPN will generally discriminate and adapt to non-linear relationships in the training data. For example, a BPN can learn the exclusive-or relation, which a single-layer perceptron cannot. Nodes in the BPN are often called "units."

The basic premise of the back-propagation algorithm has been independently derived several times. Werbos (1974) is generally credited with the first publication of the learning rule, which he called dynamic feedback; Parker (1985) gave the rule the name, "learning logic;" and the popularity of the BPN architecture is primarily attributable to Rumelhart, Hinton, and Williams (1986), as related by Simpson (1988).

All units in the BPN operate in basically the same manner. There are some slight differences dependent upon whether a unit resides in the input, hidden, or output layer. Generally, however, a unit generates an output signal as follows:

$$o_j = f(\text{net}_j) \text{ (Eq. 3)}$$

where  $j$  is a unit in the BPN,

$\text{net}$  represents the input to a unit,

$f$  is a sigmoid function,

and  $o$  is the output of a unit.

$$\text{net}_j = (\text{Sum over } i \text{ of } (o_i w_{ij})) \text{ (Eq. 4)}$$

where  $i$  is a unit in a preceding layer of the BPN,

and  $w$  is a connection weight linking two units.

$$f(\text{net}_j) = 1 / (1 + e^{-(\text{net}_j + \text{theta}_j)}) \text{ (Eq. 5)}$$

where  $\text{theta}$  is the bias weight for the unit.

An input unit has a net input which is simply the provided external input, as there are no preceding layers.

Figure 3 demonstrates a three layer BP network. At the bottom are five input units, which receive their activation from an external source. In the middle are sixteen hidden units, which receive their input

according to Equation 4. The hidden units' output is sent on over another set of weights to the single output node.

The error between the external training signal and the output node's output activity provides the basis for correcting the behavior of the network. This raw measure of error is used to find the delta for the output node.

$$\text{deltak} = (t_k - o_k) f'(\text{netk}) \text{ (Eq. 6)}$$

where k is an output unit,

t is the expected output,

f' is the derivative of the output function.

Figure 3

Fortunately, the derivative of the sigmoid function is symbolically easy to specify:

$$f' = f(1 - f) \text{ (Eq. 7)}$$

The network then must distribute this error measure backward through the network. For each of the hidden units, then, deltas are found as follows:

$$\text{deltaj} = (\text{Sum over k of } (\text{deltak} w_{jk})) f'(\text{netj}) \text{ (Eq. 8)}$$

where k is a unit in the succeeding layer of the BPN.

Each of the weights in the network is changed according to:

$$(\text{Change in } w_{ij}) = L * o_i \text{deltaj} \text{ (Eq. 9)}$$

where L is a constant representing the learning rate for the BPN.

Theta values for nodes are treated in the same manner, thus

$$(\text{Change in } \theta_{ij}) = L * f(\theta_{ij}) \text{deltaj} \text{ (Eq. 10)}$$

So, for the example BPN in Figure 3, an input vector causes output of the input units to be distributed to the hidden units, modified by the intervening weights. Similarly, the hidden units send on their output through weights to the output unit, which provides the response of the network to the particular input vector. This is called "feed-forward" processing.

Once the network output is known, it can be compared to the expected output, and the delta value for the output unit is determined. This begins the "back-propagation" process. The delta values for the hidden units can now be derived as in Equation 8. The amount of change for each of the weights between the hidden and output layers can now be found according to Equation 9. Weight changes between input and hidden layers proceed similarly. For each node in the hidden and output layers, the value of theta is also changed. This completes the normal back-propagation phase.

In the example problem, we have made a slight change to the normal back-propagation process, and have allowed theta for the input units to also be adaptively changed.

In practice, one would normally construct a network with the correct numbers of input and output units, make some guess as to the number of hidden units needed, and assign random values to the weights. The network would then be trained upon the available data vector pairs until the error

becomes suitably low, or the implementor decides to make a change in network design.

Possible applications for BPNs include encrypting, data compression, non-linear pattern matching and feature detection. Existing BP applications include translation of text inputs into phoneme outputs, acoustic signal classification, character recognition, speech analysis, motor learning, image processing, knowledge representation, combinatorial optimization, natural language, forecasting and prediction, and multi-target tracking. BP has been implemented or theorized in electronic, VLSI, and optical formats (Simpson 1988).

#### Adaptive Resonance Theory 1

Adaptive Resonance Theory 1 (ART 1) is a model introduced by Gail Carpenter and Stephen Grossberg (Carpenter and Grossberg 1987a). There are two ART models of note, ART 1 and ART 2, and many modified architectures which are premised upon one or the other of the ART models. Basically, an ART architecture is a two-layer network which provides unsupervised learning of categories of inputs (Figure 4). The F1 layer is composed of "feature nodes," which accept external inputs. In ART 1, these inputs are binary patterns, while ART 2 incorporates preprocessing to accept analog inputs to the F1 layer. The F2 layer is composed of "category nodes," which compete to respond to valid F1 activations. There are control structures built into the architecture to prevent F2 activation without input being received at the F1 layer. There are other control structures to prevent "resonance" from occurring when the prototype pattern determined by the most active F2 node does not correspond to the pattern of F1 activation.

Short term and long term memory are represented in ART architectures by node activations and inter-node weights, respectively. A "bottom-up activation" refers to the pattern of activation received by F2 nodes through the weighted links from F1 nodes. Similarly, a "top-down activation" is the pattern of activation received by F1 nodes through weighted links from F2 category nodes. Long term memory is changed only through resonance between the F1 nodes and a selected F2 node. A more Figure 4

detailed diagram of the ART 1 interconnections is shown in Figure 5.

Resonance is a state in which long term memory traces between the F1 and F2 layers are modified to more closely represent the input activation in the category node's top down weights. An F2 node which wins the competition among F2 nodes has its top-down activation tested against the bottom-up activation of the F1 feature nodes. If these match within a level of tolerance, called the vigilance level, a resonant state is entered and long term memory is changed. If not, the current winning F2 category node is made ineligible for further consideration against the input, and F2 competition is restarted among remaining eligible F2 nodes. For an example pattern presented to the ART network given in Figure

4, the presence of input turns on the gain control nodes and activates the F1 layer. The activation of the F1 layer, or bottom-up activation, is fed across a set of long term memory weights to generate a new pattern of activity at the F2 layer. The F2 layer responds with a top down activation which is filtered by the top down weights linking each of the F2 nodes to the F1 layer. Competition among the F2 nodes results in a single winner for application to the current feature input. A comparison of the bottom up activation with the top down activation yields a set theoretic measure of the match between the presented pattern and the category represented by the F2 node. If  $I$  represents the input pattern,  $V(J)$  represents the F2 category node  $J$ 's archetypal pattern, and  $p$  represents the vigilance parameter, then the cardinality of the set intersection between pattern  $I$  and the category pattern  $V(J)$  must be greater than or equal to  $p$  times the cardinality of  $I$ , or reset occurs. If reset occurs, the F2 node  $J$  becomes ineligible for matching to the current input pattern, and the process of bottom up activation, top down activation, competition, and match testing continues until some category node is found to match the input sufficiently well, or until all category nodes have been matched against the input and found to be too different. Figure 5 displays a more detailed ART 1 network, with the various weights, nodes, and connections visible.

Obviously, it is possible that none of the eligible F2 nodes will match within the vigilance level's acceptable tolerance. When an ART network is first trained, there are no eligible F2 nodes, rather there are a number of uncommitted F2 nodes. An F2 node will be selected and enter a resonant state, providing the first category. If a subsequent input does not match this category within the vigilance level, the single F2 category is rendered ineligible and the F2 layer is reset. This brings the network to a state analogous to that of having no eligible category nodes available, and a second category node is selected and resonated with the F1 layer. At any point where no further eligible F2 nodes exist, but an uncommitted F2 node remains, then a new category node is formed from the formerly uncommitted node. If no uncommitted category nodes remain, then the input has been found not to match the available categories.

Figure 5

## Chapter 2

# Integration In Neural Network Modelling

Integration in neural network modelling is taken here to mean the combination of different neural network architectures in a coordinated system. This differs from the casual usage sometimes found in the literature where the term has been applied either to systems composed of multiple units of the same base architecture, or else to trivial modifications of a known architecture. Integration applies properly to cases of multiple-architecture systems and there are some instances of systems for which the term genuinely applies but has not been used. As an example, Matsuoka, Hamada, and Nakatsu (1989) have proposed an architecture for phoneme recognition that subdivides the hidden and output layers of a back-propagation network in order to enhance the network's ability to recognize phonemes and also to substantially reduce the training time necessary for the network. However, Matsuoka terms this architecture the Integrated Neural Network (INN). While there is a substantial improvement in training time, there is no fundamental difference between an INN and a back-propagation network: they differ only in the connectivity of the weights between the hidden and output layers. The reduction in internal complexity of the INN can explain the decreased training time discovered.

A slightly different form of integration is pursued by Foo and Szu (1989). A "divide and conquer" approach to problem solving employs the same architecture, a modified Hopfield-Tank network, to handle smaller subproblems and then brings together the resulting subproblem solutions into an overall solution. This requires some coordination to effect the overall solution, bringing into play elements of the broader integrative issues I have noted, but is not properly an integrated system as I have defined it.

A better example of an integrated artificial neural network system appears in Cruz et al. (1989). Cruz uses a MADALINE architecture for image preprocessing and a back-propagation network for removing image distortion. The MADALINE architecture is a Multiple ADALINE system, developed by Widrow (Widrow, Pierce, and Angell 1961). The ADALINE, short

for Adaptive Linear Neuron, is a neuronal model using the Least Mean Square learning rule developed by Widrow and Hoff. Widrow and Winter (1988) have updated the learning rule used for multiple layer, multiple ADALINE networks. The specific example given by Widrow and Winter presents a MADALINE network for invariant pattern recognition. The MADALINE architecture most closely resembles the Perceptron architecture given by Rosenblatt. In creating his integrated system, Cruz applied the "divide and conquer" concept not for the purpose of reducing simulation time, but in consideration of space requirements needed for a system which could handle the 256x256 pixel images used.

#### Integration and Convergence

A perennial problem for the artificial neural network modeller is the issue of convergence in finite time. It is nice to know that the architecture selected for a function will converge to a solution before the heat death of the universe. It is similarly a concern that a system composed of subsidiary architectures will converge. This can be problematic, since general convergence theorems have not been found for several of the most popular architectures (Widrow 1987).

Back-propagation provides a particularly pointed example, since it is by far the most popularly used network, if number of papers concerning applications is taken as the criterion. The generalized delta learning rule of back-propagation has long been appreciated to be generally useful, yet significant progress in firmly establishing this usefulness in the form of theorems concerning convergence has been lacking. Sontag and Sussman (1989) provide a theorem demonstrating for back-propagation an analogous result to the perceptron learning rule: if a separation solution exists, the generalized delta rule's gradient descent will find the solution in finite time. While much has been made of a theorem by Kolmogorov (Farhat 1986), it must be conceded that Kolmogorov's theorem is an existence theorem: there is some network based upon the back-propagation architecture which will perform a mapping from  $n$  to  $m$ , but the theorem gives no clues as to what that specific network looks like.

There is some promising news concerning convergence which is of interest for building integrative ANN systems. Hirsch (1989) gives several theorems which hold that if component subnets of a neural network converge, then the network will converge. While Hirsch's theorems assume that the convergence properties of the subnetworks can be described by Liapunov energy functions, he notes, "It is more difficult to obtain convergence for cascades of systems that are merely assumed to be convergent, but without benefit of Liapunov functions or global asymptotic stability. One way of doing this is to place strong restrictions on the rates of convergence." Hirsch defines a cascade as a layered network where the output of one layer serves as the input of the next. Many integrative designs can be cast into this framework.

#### Incremental Synthesis

The process of synthesis leading to integrative artificial neural network modelling is important to the development of insights into topics of critical application, such as sensor fusion. By confronting directly the need for coherent internal use of available resources and capabilities, we are more likely to generate an understanding of fusion principles.

The synthetic approach to modelling provides a supportive environment for creating extensive systems. Just as the topic of artificial neural network modelling benefits from the interdisciplinary nature of its supporting sciences, so a synthetically derived artificial neural network system benefits from the range of problem solution approaches and features inherent in the underlying network architectures. By creating and maintaining a system of network architectures applicable to subfunctions in the problem solution, subfunction solution by a particular system component can benefit from the co-option of features normally found in different components of the artificial neural network system.

Networks Under Consideration, System Properties

Each of the three network architectures used in the example problem of melodic composition has its own set of features and drawbacks which play an important role in system design.

Hopfield-Tank Networks

As mentioned earlier, the Hopfield-Tank architecture is generally used in achieving a specific function. By this I mean that each Hopfield-Tank network is designed for a particular purpose, and can provide no functionality for other unrelated purposes. So the use of a Hopfield-Tank network should generally be reserved for functions which do not change over time. However, I will immediately cite a counter-example, due to its elegance of integrative design.

An ingenious mechanism for extending the utility of the Hopfield-Tank architecture is pursued by Tsutsumi (1989), where one back-propagation net remaps Hopfield-Tank network inputs and another back-propagation network remaps Hopfield-Tank network outputs. The problem given is one of avoiding robotic arm deadlocking. The movement space is constrained, and therefore applicable to Hopfield-Tank network solution. However, the adaptation of the internal space representation to the real world arm movement must be adaptive. The Hopfield-Tank network does not provide learning rules, so the back-propagation networks provide the adaptation to real world feedback. In this manner, some significant benefits accrue to the use of the integrated system: since the Hopfield-Tank network is static with respect to the encoded weights, it provides a good repository for the robot's joint-arm space; since the back-propagation networks are adaptive, the system can configure itself to respond to a changing environment.

Since the Hopfield-Tank network was conceived of in the context of implementation in silicon, the possibility for reducing a Hopfield-Tank

network instance to a hardware component can be important for real-world applications. This step would bring the benefit often touted for Hopfield-Tank networks, speed. Speed is rarely noted in practice, since most practice involves simulation. The simplicity of the Hopfield-Tank architecture can be a strong point for system design and integration, however, even in simulated systems.

A drawback which may eliminate the Hopfield-Tank network from consideration for a particular function is that the weights for the network must be derived from the constraints to be implemented and from any data functions necessary for solution but not available in the input to the network. This requires an understanding of the system to be solved, which may not be available. Some architecture with learning rule would then be more suitable for application.

The output of a Hopfield-Tank network must be deciphered from the final pattern of activation of the net at equilibrium (cf. discussion in Chapter 1). In the case of the Travelling Salesman Problem, position of active nodes provides an encoding of placement of cities in the tour. This information might be rendered more compact in another format, depending upon the input type expected for further networks in the system.

#### Back-propagation Network System Considerations

The back-propagation architecture provides a mapping from the input vector to the output vector, and can be trained by example. The system properties of back-propagation networks include stability of learning given a fixed universe. By implication, the learning is not stable if some perturbation in the problem set changes the mapping function. The back-propagation network would then learn the new mapping over time, and the old mapping would be lost.

Back-propagation networks have a moderately complex structure. The properties gained from this increase in complexity over the Hopfield-Tank architecture include the capacity for learning, the ability to extract features from input data and generate internal representations for those features, and the possibility of complex input to output transforms in accordance with learned associations.

Back-propagation networks can accept binary or analog inputs, so the inputs can represent conditional probabilities as well as more strictly constrained values. The outputs are analog values which can be interpreted as binary through the use of thresholding functions. This allows a wide variety of input and output possibilities to achieve overall system function. However, the choice of representation of inputs can be critical for speedy and reliable training to occur. For example, use of analog values should be avoided when there exists a natural partition of the range of the input into distinct states. Our example of the melodic note generator will illustrate this concern.

#### ART 1 System Properties

Adaptive Resonance Theory 1 architectures provide unsupervised



learning of "clusters" or classifications of input vectors. An internal representation of classification archetypes is generated. This architecture ensures that new inputs will tend not to perturb classifications of previous inputs. This compromise in the stability-plasticity tradeoff (Carpenter and Grossberg 1987a, b) can be modified for special purposes, as the melodic note generator program will demonstrate. The internal operation of the ART 1 network can provide certain features of especial interest to system design. Specifically, one by-product of the classification algorithm is the detection of novelty, which will be shown to have functional significance beyond that of the original design of the ART 1 network.

Some properties of ART 1 require particular attention from the designer. For example, there is a matching parameter (vigilance) which controls how much deviation from a category prototype is acceptable. There are no guidelines for the selection of the vigilance parameter, and it is left to the designer to select and assign a "proper" value. Some guidelines do exist for certain of the other learning parameters in the ART 1 architecture, such as the learning coefficients for each of the top-down and bottom-up memory equations (Carpenter and Grossberg 1987a). The ART 1 architecture provides no standard output. The designer must access internal values of the ART 1 network to provide useful information to the remainder of the system which includes that network. ART 1 is a highly complex architecture with many parameters to be selected and set by the designer. Useful modifications to be made to the architecture would include creating adaptive functions to replace some of the static and arbitrary parameters of the network, such as the gain and reset parameters (cf. Figure 5).

#### Table 1. System Properties Summary

Hopfield-Tank network properties: - Data initialization process only place for changing adaptively (not "learning" at all) - Fast convergence (on system, not necessarily on simulation) - Inflexible structure (individual design necessary) - Simple structure

Back-propagation Network Properties: - Stable learning given fixed universe - Change to design implies relearning necessary (costs for self-adaptation include forgetting what has been learned) - Adaptive weights changed as consequence of training (supervised learning) - Medium complexity of structure - Input to output transform can be computationally complex - Output nodes may deliver digital, bipolar, or analog values

ART Network Properties: - Stable self-organizing learning - Change in design produces unknown effect on existing "knowledge" - Non-adaptive parameters for gain and reset possible drawbacks, should be replaced by adaptive functions

#### System considerations

Considerations of interfacing outputs of one model to inputs of another: Inputs: HTN : analog or discrete value, one per node BPN : analog or discrete value, one per input node ART 1: discrete value, one per input node ART 2: analog value, one per input node

Outputs: HTN : discrete value, one per node BPN : analog or discrete value, one per output node ART 1 : none specified ART 2 : none specified

# Chapter 3

## Example Problem

In developing a simulation to test out processes of integration, several factors have to be considered. The simulation should have enough scope to provide good subsidiary roles for the component functions, it should be small enough to be implementable in a reasonable period of time, and it should produce output of a form which is readily apprehensible and analyzable. The first criterion is easily fulfilled; the second and third are rather more difficult to fulfill.

As a starting point for exploring the possibilities of an integrated approach to artificial neural network modelling, the problem of producing a melodic line in music composition was selected. The complexity of music composition in general provides ample considerations for the application of component networks. Unfortunately, the complexity of musical composition offers no hope for the relatively simple design and implementation of an artificial neural network system which addresses all the salient points. Therefore, simplifying assumptions are made to ease the requirements for the ANN system. The output, to be interpreted as a sequence of musical notes, can present some problems with evaluation, due to the qualitative context of musical evaluation in general. However, it is possible to treat the note sequence as a set of concatenated symbols, and apply some of the information theory concepts of Shannon (1948) to conduct an analysis.

### Simplifying Assumptions

The great complexity of musical composition in general is constrained to yield a problem of suitable scope for the example integrated ANN system. A limited scale covering one octave is assumed, in the key of C. There are no accidentals, and there is no explicit timing of notes. A single voice is assumed, and there are no harmonics generated. A limited and fixed set of classical composition rules forms the basis for the constraint and comparison of the system.

### Problem Approach

The use of several ANN architectures in creating an integrated system whose function fulfills the requirements of the musical composition

problem is assumed. A preliminary set of hypotheses as to how a composer develops a line of melody was advanced for defining the subfunctions of the composition system, or note generator. As Teuvo Kohonen (1989) notes in discussing his own ANN system for musical composition,

It is not possible to survey here the development of ideas in computer music. One of the traditional approaches, however, may be mentioned. It is based on Markov processes. Each note (pitch, duration) is thereby regarded as a stochastic state in a succession of states. The probability  $Pr = Pr(Si | Si-1, Si-2,$

$Si-1, Si-2, \dots,$  is recorded from given examples. Usually three predecessor states are enough. New music is generated by starting with a key sequence to which, on the basis of  $Pr$  and, say, the three last notes, the most likely successor state is appended. The augmented sequence is used as a new key sequence, and so the process generates melodic successions ad infinitum. Auxiliary operations or rules are necessary to make typical forms (structures) of music out of pieces of melodic passages.

Leaving the problem of constructing musical forms for possible later consideration, the approach given by Kohonen matches well the procedure of composition undertaken here. A candidate note and note sequence are proposed, a critique according to classical rules is made concerning the last note in the candidate sequence. The approval of the critic should mostly cause the acceptance of the candidate note, but the rules should be broken often enough that there is not an absolute conformity to the rules postulated.

Now we must confront the problem of how to best combine models in a unified structure that accomplishes the example function of melodic composition. The solution involves identifying the strengths and weaknesses of the component models, and which functions each may accomplish. Then when the subproblem mapping has been accomplished, it must be determined how to integrate the subfunction module outputs to other modules to accomplish the overall task.

#### Example Problem Network System

The identifiable problem subfunctions are candidate note generation, sequence critique, and novelty detection. The candidate note generator subfunction should propose notes in general, but not complete, conformance to probabilities of the next note filling the requirement of being part of a classical sequence. The sequence critique subfunction should evaluate the proposed next note in strict conformance to the set of classical sequences provided. Since the evaluation given by the sequence critique subfunction is an incomplete criterion for composition given expectations of novelty, some means of detecting novel sequences must exist for the use of the coordinating system. The coordinating system then has the information necessary to "break the rules" when needed to avoid long, boring sequences of strictly mechanical melody. This can also be seen as a requirement for stable and continued operation, as there exist sequences which have no classical next note possibilities.

#### Candidate Note Generation

The candidate note generator should produce plausible next notes given a historical partial sequence. Since the rules for identification of plausible next notes are fixed and known, there is no need for learning in this stage of the network. The candidate note generator should also occasionally provide notes which do not necessarily conform to the expectation of a classical next note.

The Hopfield-Tank network (HTN) was selected as the candidate note generator since it provides the above features. HTNs are noted for their utility in constraint satisfaction and other optimization problems, which fits the requirement that only a single next note is to be proposed at a given time and that it should basically follow the probabilities of a classical next note. A well known attribute of the HTN architecture is the inclusion of spurious local minima which do not represent "valid" solution states. By purposefully utilizing this feature, we can convert what normally constitutes a drawback into a asset for the subfunction. The spurious local minima will give the occasional proposal of non-classical next notes. The constant and known nature of expected sequences determines the formation of the HTN weights, in conjunction with the known constraints for proper HTN function. The use of "noisy" input values can produce the semi-random distribution of possible notes that is needed for variability.

Figure 6 shows the modified HTN architecture, called Bach, used in our simulation. The rows represent note values and the columns represent sequence placement. The constraints imposed on this network include the need to present a single winning note in each place in the sequence pattern, and the need to prevent endless repetition of the same candidate note. By introducing relatively strong inhibitory links within rows and columns, we can satisfy the constraint requirements. We achieve preferential selection of classical next notes by reducing those inhibitory links somewhat for connections which follow classical sequence patterns.

#### Sequence Critique

The sequence critic should provide an evaluation of the conformance of the proposed next note to classical melodic sequence rules. If the sequence rules are assumed to be provided by example, then some learning function is required to allow the critic to become adept and reliable. The subfunction receives a note sequence as input, and produces an output which may be interpreted as a boolean statement of the classical nature of the candidate note.

The back-propagation network (BPN) was selected as the sequence  
Figure 6

critic. The BPN, as discussed in Chapter 2, can learn any input/output transformation given to it (within some constraints upon the availability of sufficient hidden nodes to form a stable internal representation). The form of the BPN used, called Salieri, was in the end a network

accepting a binary representation of the input sequence, using twenty hidden units for internal representation, and producing an output value which was interpreted as a yes-or-no output. So the net structure used forty input units, twenty hidden units, and one output unit. This is a medium sized BPN.

#### Novelty Detection

The novelty detection function must have components to enable the recollection of past sequences for discrimination of novel sequences. However, the space requirements should not be overwhelming. A classification system would provide good data compression while allowing the nearly complete context information needed for novelty detection.

The ART 1 architecture was selected for the role of the novelty detector, for it provides novelty detection as part of a classification framework. The ART 1 architecture also has other features of interest to further research on integrated and extensive systems.

The ART 1 network used here, called Beethoven, is modified from the Carpenter-Grossberg architecture. Some of the explicit features of the Carpenter-Grossberg architecture are handled as implicit assumptions in the procedural simulation. The "2/3rds Rule," for example, is not invoked, since the only time that Beethoven is active, the network meets the constraints of the rule. The separate rules for top-down and bottom-up weight modifications are replaced by a single rule for both, as is done in the ART 2 architecture (Carpenter and Grossberg 1987b).

#### Coordination

In any integrative system, some means of coordinating subfunctions becomes necessary. Some interpretation and processing of input or output terms may be accomplished by the coordinating system. The ultimate decision for whether or not a candidate note is accepted falls to the coordinating system.

The coordinating system is called Lobes, since the features and activities of the frontal lobes (Levine 1986, Levine and Prueitt 1989) provided the inspiration for its operation. Lobes generates the context management and state dependent actions which drive the integrated system to completion of the intended function, melodic composition.

Lobes also contains an internal boredom function, which tends to increase over time. There is a boredom threshold, which causes a change in behavior in Lobes if it is exceeded.

#### Operation

In operation, the integrated ANN note generator uses its components in a sequential manner. Lobes generates a call to Bach for a candidate note, which when returned is sent to Salieri for critique. As a mechanism for preventing wastage of Beethoven category nodes, the entire sequence which Bach settles upon is used for further processing by Salieri and Beethoven. Since at the beginning of composition there is no sequence history, Bach's inputs are determined randomly over the entire sequence.

As notes are added to the sequence history, Bach inputs are determined with the addition of some noise, except for the candidate note column which receives only random noise as input.

Salieri receives the Bach sequence values and makes a determination of classical conformance, which it passes back to Lobes. Lobes then sends on the entire context, sequence plus critique, to Beethoven. At first, Beethoven is virtually certain to encode new categories for each input it receives. As time goes by, the likelihood that Beethoven will encode a new category decreases, until all category nodes are utilized. So it is more likely that the first few notes generated will conform to classical sequence examples. Sometimes, however, a note which has no possible classical successors will be generated early in the simulation. In this case, it is likely that Beethoven's category encoding will proceed at a much faster than average pace.

The indication of novelty generated by Beethoven is used by Lobes to modify the system response to the internal state, which is determined by Salieri's critique of the next note, Beethoven's detection of novelty, and the boredom threshold in Lobes. If Lobes is not bored and Salieri approves of the candidate note, the note is accepted. If, however, Lobes has reached its boredom threshold and Salieri approves of a note, it will reject the candidate note and request another one from Bach. Likewise, if Salieri disapproves of the note but Lobes is bored, Salieri may be overridden and Lobes may accept the note. Indications of novelty from Beethoven can satisfy Lobes' drive to be "excited," or not-bored. This will tend to make Lobes more conservatively classical as Beethoven continues to detect novelty. Figure 7 shows the operation of the coordinated system.

Figure 7

# Chapter 4

## Results

### Performance

The integrated ANN note generator system produces 152 notes in about three hours when run on an 80386 PC compatible at 16MHz clock speed. It takes approximately fifteen hours to produce the same number of notes on an 8088 based machine with an 8087 numeric coprocessor.

### Example and Analysis of Output

Appendix A contains sample output from a run of the note generator.

With a problem such as musical composition, assigning an objective measure denoting the "worth" of the output is not possible. However, it is possible to compare the output of our note generator system with random sequences of note values. By use of a binomial performance measure, it is possible to define how much the sample output differs from a random sequence.

Random sequences have their own mystique and interest, but subjective evaluations of random melodic forms by the untrained ear tend toward the negative. The output of our note generator network was intended to basically follow the guidelines of an example set of classical sequences.

The system included mechanisms for breaking out of a strict adherence to the guide set of sequences. Thus, it would be expected that the output have somewhat more resemblance to random sequences than the "rules" would state. Since the classical rules of composition, while not our sole criteria of fitness, are the only quantifiable part of our criteria, we compare our output with random note sequences on the basis of these rules.

Table 2 summarizes the characteristics of output sequences generated under several different conditions. A random note generator, a mostly classical note generator, and our integrated ANN note generator each produced outputs and were evaluated using the critic developed for training Salieri. The "Successes" column indicates the number of times the next note in the sequence could be considered to be classical.

The random note generator simply output a sequence of random numbers for a set sequence length. The Classical Instructor sequence generator operated by determining the available pool of possible classical next

notes, then randomly selecting one of those notes. In some cases, no next note fit the criteria of being "classical," and a random note was generated. The rest of the sequence generators were variations upon our ANN note generator. The use of different back-propagation nets in the critic role gave different results in the output. Trained and untrained back-propagation nets with analog inputs were used. No significant difference in output could be distinguished between the trained and untrained versions, but the result was far closer to random performance than classical. The inability of the Salieri net with analog inputs to converge to a reliable and accurate performance measure explains the similarity of result with the untrained version of the same type. On the other hand, a Salieri composed of a trained back-propagation network using a binary representation for inputs was able to converge to a stable and fairly accurate state. Hence the performance of the trained Salieri with binary inputs was considerably closer to classical than was the performance of the random sequence generator. In two cases, a rule-based critique system was substituted for the back-propagation network.

Table 2. Classical components of output sequences.

Sequence generator	Sequence	Successes	% Z	versus Z	versus Length	random	classical
Random	10,000	890	0.089	0.00	-102.54		
System w/ untrained Salieri (analog inputs)	152	18	0.118	1.26	-23.84		
System w/ trained Salieri (analog inputs)	152	19	0.125	1.54	-23.64		
System w/ trained Salieri (binary inputs)	152	36	0.237	6.28	-20.07		
System w/ rule-based critique (Salieri supervisor)	Run 1	150	35	0.233	6.10	-20.06	Run 2
	150	47	0.313	9.42	-17.50		
Classical Instructor	8,150	6898	0.846	102.54	0.00	(rule-based critique w/out ANN system)	



# Chapter 5

## Discussion

The integrated note generation system's performance suggests that it met operational expectations. It produced notes according to a mixture of somewhat conflicting criteria. This kind of operation in the midst of uncertainty characterizes many human decision-making processes, and may be assumed to play a role in human music composition as well. Our framework allows for further experimentation with hypotheses concerning the fundamental processes involved in higher-order constraint satisfaction systems in an extensive environment (cf. Pao 1989).

The integrated approach has demonstrated several advantages and disadvantages in development and operation. The disadvantages include the complexity of handling several different network architectures at once, which can contribute to programmer confusion (the downfall of programmer omniscience). The necessity for dealing directly with "model mismatches," where one subnetwork may produce a different representation as output than the next subnetwork requires as input, can cause system design time to be protracted. Failure to recognize that a problem exists in articulation of networks can result in behavior that diverges wildly from expected norms. On the other hand, any complex problem may present similar difficulties regardless of the choice of solution approach. By using subnetworks of known characteristics, one may be able to achieve a solution with fewer uncertainties than a totally top-down approach would yield. With a range of different capabilities available from subsidiary networks, the likelihood of encountering an insoluble subproblem is reduced. Synthetic integrated systems also lead to a combinatorial explosion in the richness of possible system behaviors, which again is reminiscent of the increasingly interesting behaviors noted as more complex biological organisms are considered.

### Simulation Concerns

The integrated ANN system from the example problem relied on several procedural programming shortcuts. For example, the implementation of "boredom" in Lobes was simply a counter which would be compared with a threshold value. This does not have any basis in biological neural

systems, yet rather neatly simulates the behavior of a simple neuronal model for the same task. As another example, the decision in the ART 1 network as to which category an input belongs to is currently made on the basis of an arbitrary, winner-take-all rule. The same effect could probably be achieved by means of on-center-off-surround neural interactions that are more biologically realistic. Casting the system functions into an entirely biological framework would yield a better, more capable system for future work. However, the system stands as a first effort toward this goal.

#### Integration and Artificial Intelligence

There have been several efforts toward integrative system design in the top-down modelling school. In the HEARSAY system (Reddy et. al. 1973), the blackboard model was developed. The blackboard model presupposes the combination of a set of possible problem-solving systems which have available the current system state. The system state is said to reside upon "the blackboard" as a visualization of the process of problem-solving. Each of the several applicable subsidiary problem-solving systems attempts to derive an incremental step toward the global solution, and competes with other such problem-solvers to control the blackboard, and thus be able to change the system state. This is a significant development, and one which is paralleled by concepts in several artificial neural network models. In ART models, for example, the concept of competition among various classification prototypes bears a strong resemblance to the blackboard model. If the F1 layer's activation is considered analogous to the system state, the F2 nodes each are analogous to problem-solvers in the blackboard model.

There have been some efforts toward explicitly bringing together top-down and bottom-up models in hybrid systems. Amano et al. (1989) present an example of a phoneme recognition system combining an expert system with a perceptron network. The expert system provides feature extraction from speech data, which is input to the perceptron. The perceptron allows decision-making under uncertainty, whose output is interpreted using fuzzy logic rules. This avoids drawbacks associated with "template matching" phoneme recognition schemes. Rabelo and Alptekin (1989) have integrated a neural network with an expert system into an intelligent scheduling system for manufacturing applications. Their system has the ability to learn from experience and generate schedules within real-time constraints.

#### Neurobiological Evidence of Multifunctionality

Multi-state functionality of memory is supported by work of Nottebohm (1989). Nottebohm's work involves the development and remembrance of complex songs in songbirds. Through a series of studies, Nottebohm demonstrates that hormonal changes can cause the forgetting of songs in a bird's repertoire, or allow the formation and remembrance of new songs. The hormonal changes in question center around testosterone, and typically the mating season is the time when levels of testosterone allow the

formation of songs, presumably conferring a reproductive advantage for male songbirds. Nottebohm demonstrates that the changes are only hormonally dependent by the simple strategy of artificially inducing song creation by the application of testosterone to songbirds of both sexes at various times of the year. The withholding of regular doses of testosterone was also shown to cause the forgetting of known songs in the same birds.

The implications of Nottebohm's work include support for state-dependent memory. Since a specific memory function can be modulated by a specific hormone, this implies that other memory systems may also have recall dependent upon some hormone or other chemically mediated process.

Given that recall and learning can be so modulated, the necessity for taking state dependencies (levels of hormones and other neuroactive substances) into account for system function can be appreciated.

State-dependent memory is also supported by the work of many other investigators, such as Bower (1981). In Bower's experiments, happy or sad moods were induced in subjects by hypnotic suggestion, in order to investigate the influence of emotions on memory and thinking. This influence was profound; for example, people recalled a greater percentage of those experiences which had taken place when they were in the same mood as they were in during recall. Also, when the feeling tone of a story agreed with a reader's hypnotically generated emotion, the reader found the events and characters in the story more memorable and easier to identify with.

State dependence of memory or other neural function can give rise to quite useful modelling constructs. The ability to recast problem solutions given functional states becomes biologically justifiable. The logical power of conditional activation of entire subnetworks becomes available through the modelling, however coarse, of these state dependencies. The almost direct implementation of expert system analogues which can be analyzed in a completely biological and ANN context is made possible.

Implications include the higher order integration of functionally changed sub-units over time. In humans, well known state dependencies include the fight or flight response to norepinephrine production and the diving response typical of mammals entering cold water. Without the appropriate integrative control, neither fight or flight nor the diving response would produce the desired, or selected, effect. The coordination of separate functional neural "circuits" is clearly present; the exact mechanism remains to be elucidated but there have been some promising beginnings in neural network models. For example, in the neural model of attention described by Grossberg (1975), there is competition between nodes representing activations of different drives (hunger, thirst, sex, etc.). The winner of this competition is not determined solely by which drive is highest, but also by the availability of compatible cues in the

environment.

State-dependent memory implies the existence of functional changes over time in cortical structures. Since we now have evidence of multi-modal neural circuitry, at least some consideration should be given to the implied necessary integration. The problem of understanding a system which is dynamic not only in processing of input but also in functional neural subsystems is both daunting and exhilarating. It is daunting, because the complexities of modelling such systems exceeds our current capacity for ready assimilation and understanding of the underlying concepts and mechanisms (which have not yet been elucidated), and exhilarating because there appears to be no end to the variety of expression of these systems in the natural world, and thus no apparent end to the problem-solving challenge awaiting the researcher.

The function of speech processing in humans, for example, requires the acquisition of external signals, the separation of those signals into semantic and affective content, the recognition of mode in affective content, the parsing of semantic content, and the integration of semantic and affective content to determine meaning. This list of subfunctions is not complete, which gives an indication of the extent to which integration remains a regular and important activity in biological systems.

#### The Triune Brain Theory

Integrative theories of neural/cognitive function have a long history. One of the best known is the triune brain theory of Paul MacLean (1970). MacLean's research into behavioral studies of different brain areas led him to propose that the human brain is divided into three developmentally derived regions of separate function (see Levine 1990 for discussion). The earliest, and presumably the most primitive region, is termed the reptilian brain, and is composed of the the brainstem and basal ganglia. The reptilian brain is responsible, in this theory, for the preprogrammed, innate behaviors. The paleomammalian brain, composed of the limbic system, modifies the expressed pattern of reptilian brain responses and is the source, in this theory, of the basic emotions (love, hate, fear, arousal, etc.). The neomammalian brain, composed of the newer parts of the cerebral cortex, provides further modifications of the expression of the two older brain areas, and gives us our rational capacity, seen in the ability for planning and verbal expression.

While MacLean's theory is oversimplified, it does provide a useful set of distinctions between various cognitive subfunctions, all of which are involved in complex behaviors. In fact, if one stretches the imagination, one can draw analogies between the reptilian brain and our Hopfield-Tank network; the paleomammalian brain and our ART 1 network, and the neomammalian brain and our back-propagation network.

The integrated ANN note generator had its origins in a collaborative effort to develop an extensive ANN system suitable for exploring multi-modal cognitive hypotheses (Blackwood, Elsberry, and Leven 1988). That

project, in turn, was derived from insights provided by Leven (1987b). Leven's SAM model was depicted in a manner which led to a discussion of the possibility of replacing the components of MacLean's triune brain model with current ANN architectures. The difficulty of describing a suitably restricted problem for adequate application of the limited current architectures was resolved with the simple melodic composition problem outlined previously. Points of difference from the original MacLean theory can be attributed to certain changes in model context (as modified by Leven's separation of memory into three components: motoric/instinctive [reptilian], sensory/affective [paleomammalian], and associative/semantic [neomammalian] (Leven 1987a)) and to the mismatch between architectures derived not for their similarity to these basic cognitive forms, but to satisfy more immediate criteria such as being implementable in current electronic devices. The desired system based on our loose analogy to MacLean's theory has been demonstrated to be operational and ready for incremental refinement. We hope that in future work such analogies can be made more precise. The development of both neural network theory and neuroscientific data should allow the critical research to continue into these theories of integrative cognitive function.

# Appendix A

## Sample Melody Output Of The Various Note Generator Programs

Integrated ANN Note Generator Sample Output

b61T output, page 1

b61t output, page 2

b61t output, page 3

Random Note Generator Sample Output

random output, page 1

random output, page 2

random output, page 3

Classical Note Generator Sample Output

classical output, page 1

classical output, page 2

classical output, page 3

## Appendix B

# Program Source Listing: Integrated Ann Note Generator

This appendix is represented in the repository by the legacy source and data files in THES/. The automated thesis conversion suppresses the full listing here to keep the document manageable.

# Appendix C

## Program Source Listing: Back-Propagation Unit

This appendix is represented in the repository by the legacy source and data files in THES/. The automated thesis conversion suppresses the full listing here to keep the document manageable.



## Appendix D

### Program Source Listing: List Structures Unit

This appendix is represented in the repository by the legacy source and data files in THES/. The automated thesis conversion suppresses the full listing here to keep the document manageable.

## Appendix E

### Program Source Listing: Miscellaneous Procedures Unit

This appendix is represented in the repository by the legacy source and data files in THES/. The automated thesis conversion suppresses the full listing here to keep the document manageable.

## Appendix F

### Program Source Listing: Global Type And Variables Unit

This appendix is represented in the repository by the legacy source and data files in THES/. The automated thesis conversion suppresses the full listing here to keep the document manageable.

# Appendix G

## Program Source Listing: Classical Instructor Unit

This appendix is represented in the repository by the legacy source and data files in THES/. The automated thesis conversion suppresses the full listing here to keep the document manageable.

## Appendix H

### Program Source Listing: Ansi Screen Control Unit Unit Ansi\_Z;

This appendix is represented in the repository by the legacy source and data files in THES/. The automated thesis conversion suppresses the full listing here to keep the document manageable.

# Appendix I

## Program Source Listing: Musical Sequence Evaluator Program

This appendix is represented in the repository by the legacy source and data files in THES/. The automated thesis conversion suppresses the full listing here to keep the document manageable.

## Appendix J

### Program Source Listing: Random Note Generation Program

This appendix is represented in the repository by the legacy source and data files in THES/. The automated thesis conversion suppresses the full listing here to keep the document manageable.

## Appendix K

### Program Source Listing: Note Sequence Playing Program

This appendix is represented in the repository by the legacy source and data files in THES/. The automated thesis conversion suppresses the full listing here to keep the document manageable.



## Appendix L

### Program Source Listing: Rule-Based Note Sequence Generation Program

This appendix is represented in the repository by the legacy source and data files in THES/. The automated thesis conversion suppresses the full listing here to keep the document manageable.

## Appendix M

### Program Source Listing: Offline Back-Propagation Network Training Program

This appendix is represented in the repository by the legacy source and data files in THES/. The automated thesis conversion suppresses the full listing here to keep the document manageable.

## Appendix N

### Data File Listing: Hopfield-Tank Network Weight Data File

This appendix is represented in the repository by the legacy source and data files in THES/. The automated thesis conversion suppresses the full listing here to keep the document manageable.

# Appendix O

## Data File Listing: Classical Sequences Data File

This appendix is represented in the repository by the legacy source and data files in THES/. The automated thesis conversion suppresses the full listing here to keep the document manageable.

## Appendix P

### Data File Listing: Back-Propagation Network Data File

This appendix is represented in the repository by the legacy source and data files in THES/. The automated thesis conversion suppresses the full listing here to keep the document manageable.

## Appendix Q

### Program Source Listing: Translator From Program Note Files To Music Transcription System Song Format

This appendix is represented in the repository by the legacy source and data files in THES/. The automated thesis conversion suppresses the full listing here to keep the document manageable.