The Spreadsheet Implemented "Creativity Machine" - Its Construction, Function, and Current Successes

S.L.Thaler
Dendrite Neurocomputing
12906 Autumn View Dr.
St. Louis, MO 63146-4331
sthaler@ix.netcom.com

Abstract *- When we subject the internal architecture of a trained artificial neural network to various forms of perturbation, the resulting series of network activations tends to frequent attractor basins representative of the exemplars the net has 'seen' in training. For instance, a network exposed to examples of culturally accepted music tends to revisit those melodies or slight variants thereof when its weights and biases are chaotically perturbed. Gradually increasing the level of such internal noise, we smoothly depart from the known domain of musical knowledge used for network training and begin to produce never before heard melodies possessing various degrees of esthetic appeal. In essence, by gradually softening all the network's constraint relations (i.e., its connection weights) governing musical acceptability, we produce orchestrations increasingly removed from known compositions. If we supervise the stream of melodies emerging from this net with a second network trained to judge musical esthetics, we create a so-called "Creativity Machine." Such a virtual machine may produce thousands of new songs in a period of hours, or, if suitably retrained, invent hundreds of candidate high temperature superconductors in a comparable time frame.*

Until recently, we have fashioned such Creativity Machines from algorithmic programming languages such as C/C++ and Visual Basic. As the machine's complexity grew, incorporating multiple chaotic networks and supervising networks, the intricacies of connecting and coordinating these nets by algorithmic code became unbearably difficult. Thus arose the need for a highly graphical user interface that could allow an operator to manually cut, paste, and link the functioning networks into position within the Creativity Machine cascade. What arose from this methodology was a technique more resembling 'knitting' than computer programming. We found that the natural medium for implementing this graphical neural network interface was a spreadsheet application, notably Microsoft Excel™, wherein we achieved the crucial network function of synaptic integration by relative referencing and appropriately weighting signals from afferent spreadsheet cells. Resident spreadsheet functions formed the basis of neuron activations, allowing a wide range of transfer functions including the usual linear, linear threshold, and sigmoidal activation schemes. Further, the clear layout of the network cascade within the spreadsheet context, color-coded neuron activation levels and spreadsheet-resident cell reference trace facilities all assisted and accelerated the construction and debugging of these complex systems.

Reflecting this flexibility and ease of design, we have realized a diverse range of Creativity Machine applications ranging from subjective regimes such as entertainment and the arts to more objective endeavors, as in the search for ultrahard materials and targeted medical treatment.

1. The Creativity Machine Paradigm

Typically an artificial neural network (ANN) interprets the introduction of some perturbation to its internal architecture as the application of some environmental or environmental-like feature (i.e., training exemplars or their generalizations) to its inputs (Thaler, 1993; Thaler, 1995; Thaler, 1996; Yam, 1995). We may view such phenomena as a generalization of the *vector completion* property of neural networks whereby missing input information to the net is 'filled in,' so to speak, by way of the collective behavior of the ANN's processing units. Similar principles would now apply to disturbances administered to various internal architectural features such as weights and biases within a trained ANN. The network collectively activates as though responding to environmental features in a process coined 'internal vector completion.' If the level of perturbation is relatively small, the network will tend to settle into an activation state representing an intact training exemplar or memory. As the magnitude of perturbation increases, however, the network may fail in correctly classifying the internal perturbation as a known training vector, activating into some novel pattern representing a false memory or confabulation. The greater the perturbation the less features the corrupt memory shares with the training exemplars. Therefore, this graduated internal perturbation process offers a technique for the generation of notions progressively removed from the body of concepts embodying any knowledge domain.

For instance, a network exposed to examples of culturally accepted music tends to revisit those melodies or slight variants thereof when we perturb its weights and biases. Gradually increasing the level of such internal noise, emerging concepts smoothly depart from the known domain of musical knowledge used for network training and the network begins to produce never before heard variants possessing various degrees of esthetic appeal. In essence, by gradually softening all the network's constraint relations governing musical acceptability, we produce musical possibilities increasingly removed from known compositions (Figure 1). If we supervise the stream of musical candidates emerging from this net with a second network trained to judge musical esthetics, we create a so-called Creativity Machine (Figure 2). In general these machines consist of at least two networks. I call the first, perturbed network an "imagination engine" (IE) and the second patrolling net an "alert associative center" (AAC). Harnessed together these two networks may produce thousands of new songs in a period of hours (Thaler, 1994), or, if suitably retrained, invent hundreds of candidate high temperature superconductors in comparable time frames.

We note that the "Creativity Machine Paradigm" is the most desirable of discovery search mechanisms representing a "multistage process" (Rowe and Partridge, 1992) in contrast to the alternative and less efficient "neo-Darwinian" or "neo-Lamarckian" systems. The former neo-Darwinian technique generates myriad unconstrained concepts, and then enlists algorithms to sift through the often impractically large collection of generated ideas. At the other extreme, neo-Lamarckian processes laboriously invent possibilities rigorously obeying the knowledge base constraints and then choose the first to emerge. Because the CM Paradigm progressively softens constraints, we may continuously range between both extremes, perhaps starting with a Lamarckian-style search using a minimally perturbed network and finally arrive at a Darwinian search at high levels of chaos. By ramping up the RMS noise level applied to the network weights and biases (the parameter ρ in Figure 2) we may arrive at a level of constraint softening that minimizes search time and maximizes the quality of emerging concepts. In effect, the technique is rather analogous to a Taylor series expansion of some analytical function about a point. Here, we are expanding some vectorialized concept around some known knowledge domain or training set using a neural network model.

Besides its inherent search efficiency, the connectionist design of the Creativity Machine obviates the need of human designers to either glean or understand the rules and models implicit within any given knowledge domain. That is, we train both the chaotic network and its supervisor by example. The self-organization of artificial neural networks to reflect the underlying schema of any knowledge domain makes discovery systems attainable within hours instead of the months or years so characteristic of expert system development (where all the why's and wherefore's must be gleaned and then incorporated into extensive "if-then" production code). As a result we may devise new Creativity Machines focused on discovery in altogether different problem domains in astoundingly short time frames. We are thus able to implement machines whose functions may vary from predicting hundreds of new high temperature superconducting oxides to recommending portfolio distribution in stock market transactions.

Similarly to the invention of new conceptual knowledge, the Creativity Machine is capable of novel movement planning and attention windowing. That is, we may train an auto-associative network on several prototypical robotic movements. This mapping likewise will contain all the implicit constraints inherent within the robotic system's allowable range of movements. By now subjecting that trained network to internal chaos, it will produce kinematics unlike its training exemplars, yet generally obeying

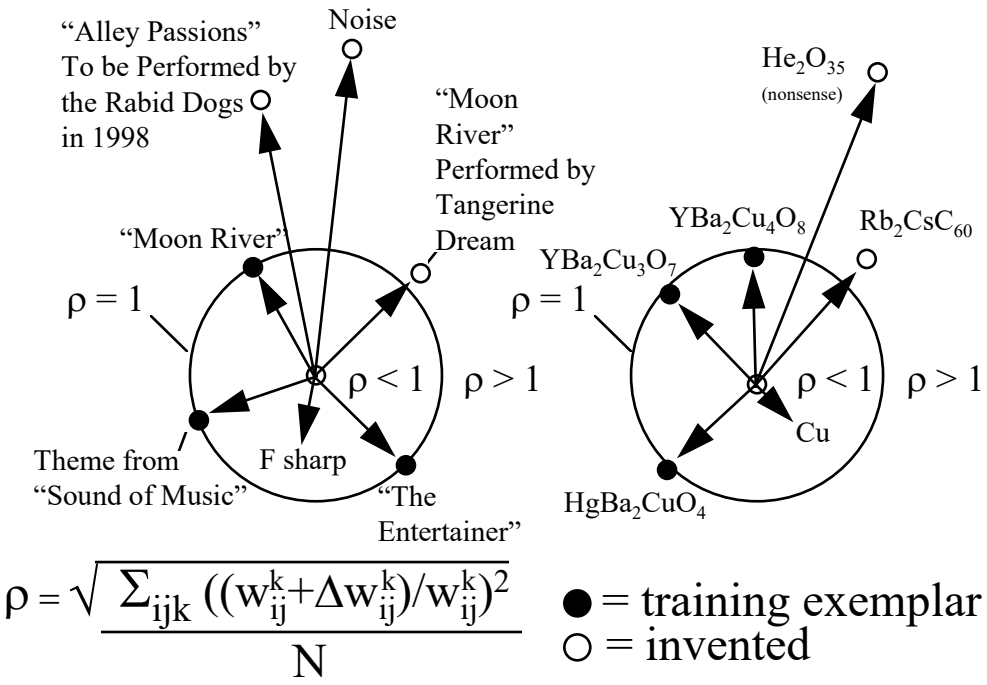


Figure 1. By gradually increasing the level of mean perturbation applied to the connection weights within an ANN trained in a given knowledge domain, we provide a search mechanism for new concepts just beyond that domain. Above, we denote the chaos parameter ρ as the RMS deviation of all weights in the network from their trained values, where w_{ij}^k represents the weight connecting the i^{th} and j^{th} neurons in the k^{th} layer and Δw_{ij}^k signifies that weight's perturbation and N is the number of weights in the net. Neo-Lamarckian search would correspond to constraining network perturbation to the unit circle (i.e., the set of all unperturbed weights), while neo-Darwinian search would consist of the myriad points inside and outside the unit circle where ρ is much different from 1. We reach a satisfying compromise by gradually varying ρ from a value of 1 in either direction (Note that we may further filter the search by the dynamic pruning of neurons generally not contributing to the desired end-concepts.).

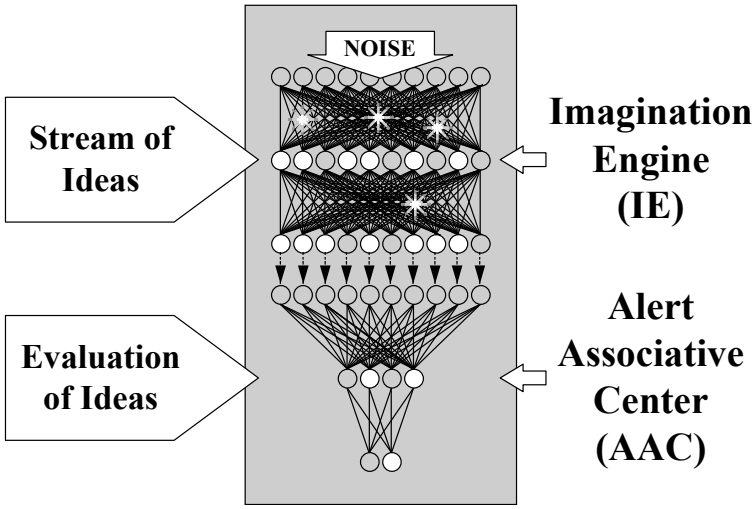


Figure 2. Fundamental creativity machine architecture consists of a chaotic network or "imagination engine"(IE) supervised by a quiescent network or "alert associative center" (AAC). The starbursts denote transient fluctuations in the values of trained-in weights that drive both the recall and degradation of stored memories.

its mechanical constraints. A second supervising network may then select a movement scenario emerging from the chaotic net, satisfying some predetermined objective (i.e., when a robot must reach around some interposed obstacle to pick up an object). Likewise, an attention windowing Creativity Machine may look at some grouping of data, taking a number of alternative perspectives. Training for such a chaotic network proceeds by showing an untrained auto-associative net several candidate viewing angles or data groupings consistent with the databot's sensing ability and problem context. In operation this CM will exhaust all possible data viewing windows and, if necessary, invent totally new perspectives.

Admittedly a powerful discovery tool in its rawest algorithmic form, we now introduce two improvements that have greatly added to the Creativity Machine Paradigm's overall potential. The first of these new twists amounts to a convenient graphical interface for the rapid prototyping, testing, and implementation of complex Creativity Machine designs. The second innovation enables such graphically rendered Creativity Machines to simultaneously learn and create without the need for operator intervention. Combined with the above capacity for novel movement and attention planning, the Creativity Machine attains the status of an 'autonomous agent.'

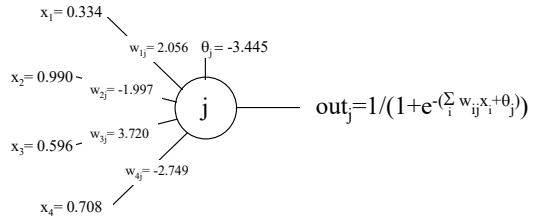
2. The Spreadsheet Implemented Creativity Machine

We typically think of neural network simulations as the sequential evaluation of activation states of neurons within a network, using some algorithmic language such as C or C++. Within such schemes, individual activation levels are only momentarily visible and accessible, as when the governing algorithm evaluates the sigmoidal excitation of any given neuron (Figure 3). Except for its fleeting appearance during program execution, a neuron's excitation becomes quickly obscured by its redistribution among downstream processing elements.

```
void feedforward(float *input, float *output)
{
        ●
        ●
        ●
        for(j=0; j<nodes[lay]; ++j)
                act[j]= 1.0/(1.0+exp(-net[j]));
        ●
        ●
        ●

}
```

Figure 3. Neuron activation within a given network layer is evaluated in C-code.



	A	B	C
1	0.334	2.056	1/(1+EXP(-(SUMPRODUCT(A1:A4,B1:B4)+B5)))
2	0.990	-1.997	
3	0.596	3.720	
4	0.708	-2.749	
5		-3.445	

Figure 4. A neuron implemented in a Microsoft Excel spreadsheet.

Exploiting the many analogies between biological neurons and cells within a spreadsheet, we may evaluate the state of any given network processing unit by way of relative references and resident spreadsheet functions (Figure 4). By referencing the outputs of such spreadsheet neurons to the inputs of other similarly implemented neurons, we may create whole networks or network cascades. Unlike the algorithmic network simulation, all neuron activations are simultaneously visible and randomly accessible within this spreadsheet implementation. More like a network of virtual, analog devices, we consider this simulation to be a distributed algorithm, with all neuron activations updated with each wave of spreadsheet renewal.

As a further benefit of the spreadsheet implementation, the user now has a convenient graphical interface for constructing and experimenting with ANNs. For instance, we need only build a template neuron once, using simple copy and paste commands to build and connect whole networks from a prototypical processing unit. We may repeat these procedures on larger scales to move networks into position and link them into larger cascade structures. The resulting compound neural networks are transparent in operation and easily accessible for modification and repair. Furthermore, the approach lends itself well to the rapid prototyping of neural architectures when faced with multiple alternative neural circuitries.

In contrast to existing neural network toolboxes that allow for the cascading of multiple neural networks, we observe that such systems generally represent a graphical interface to some underlying algorithmic source code (i.e., a DLL).

Within our ANN implementation, processing takes place strictly within the confines of the spreadsheet environment, with all processing units constantly accessible for various operator interactions and modifications. Thus, it is possible to readily involve various hidden layer neuron interactions within cascade function, to add various functional perturbations to select network weights, or to add recurrencies within any processing unit or groups of neurons. Furthermore within sophisticated spreadsheet applications such as Microsoft Excel we may enlist various resident functions and diagnostics such as the dependency traces among network cells and real-time plotting of network activation levels.

Because Creativity Machines require at least two ANNs linked within a cascade structure, the spreadsheet implementation discussed above is the most natural construction technique. We may readily connect Imagination engine outputs to AAC inputs by relative cell references. Further, we are at liberty to add various forms of functional perturbations to the constant connection weight values to optimize the generation rate of useful concepts. At all stages of construction we may enlist any of the resident Excel facilities to plot the behavior of any neuron or neuronal cluster or to perform various functional traces throughout the connectionist structure.

A Visual Basic macro resident within the same Excel workbook typically drives the spreadsheet-implemented Creativity Machine. Its chief functions are to (1) administer random perturbations to individual neurons or connection weights within the imagination engine, (2) enable any recurrencies within the Creativity Machine architecture (i.e., Excel does not allow for self-referent loops), and (3) perform any run-time diagnostics of the machine.

Perhaps the spreadsheet implementation of the Creativity Machine sees its greatest advantage when the necessary cascade structure consists of many highly interconnected networks. For instance, various juxtapositional invention schemes require the use of several imagination engines running in parallel and patrolled by a single AAC (Thaler, 1996b). Alternately, a number of independent AACs, each alert to separate judging criteria of a single IE's output, may be simultaneously active. Here, the ability to rapidly interconnect these various modular networks by relative cell reference tremendously expedites overall CM construction and debugging.

3. Autonomous Discovery / Creativity Agents

Although the Creativity Machine Architecture described above offers unique potential in the areas of computerized discovery, there still remains a significant obstacle to the complete autonomy of such systems: Since each of the component networks must be pretrained within their respective conceptual spaces, there is no facility for the ongoing learning. Therefore, the overall CM architecture cannot create new ideas based upon this expanding or evolving experience.

Serendipitously, two recent developments have made it possible to incorporate such real-time learning by spreadsheet-implemented Creativity Machines. The first of these enabling technologies is the prevalent use of Dynamic Data Exchange (DDE) to deliver newly arriving data to an Excel spreadsheet. Thus provided with the equivalent of sensory input (i.e., satellite downlinks to the stock market or signals from laboratory sensors) any fresh information becomes immediately available to any spreadsheet systems.

The second enabling technology represents a means of training any spreadsheet-implemented ANN without the use of a backpropagation algorithm. Instead, such self-training systems use a second spreadsheet network, containing a distributed backpropagation algorithm, to train the first (Thaler, 1996b). In this way, such Self-Training Artificial Neural Network Objects (STANNO) achieve seamless, uninterrupted adaptation to newly arriving data. An important capacity of the self-trainer scheme is that several STANNOs may simultaneously train within the same spreadsheet application, thereby permitting multiple perspectives on any given data base.

If we supply relevant data streams to both Creativity Machine networks, training will continuously update itself until adequately trained. The governing algorithm will then paste copies of these networks into their respective positions within the Creativity Machine architecture. Therefore, we do not need to interrupt the CM's process of invention and discovery to retrain and reincorporate these updated networks into the cascade architecture. We may then accumulate a dynamic library of CM discoveries (e.g., unsynthesized compounds and their estimated physical and chemical properties) as intimated in Figure 5.

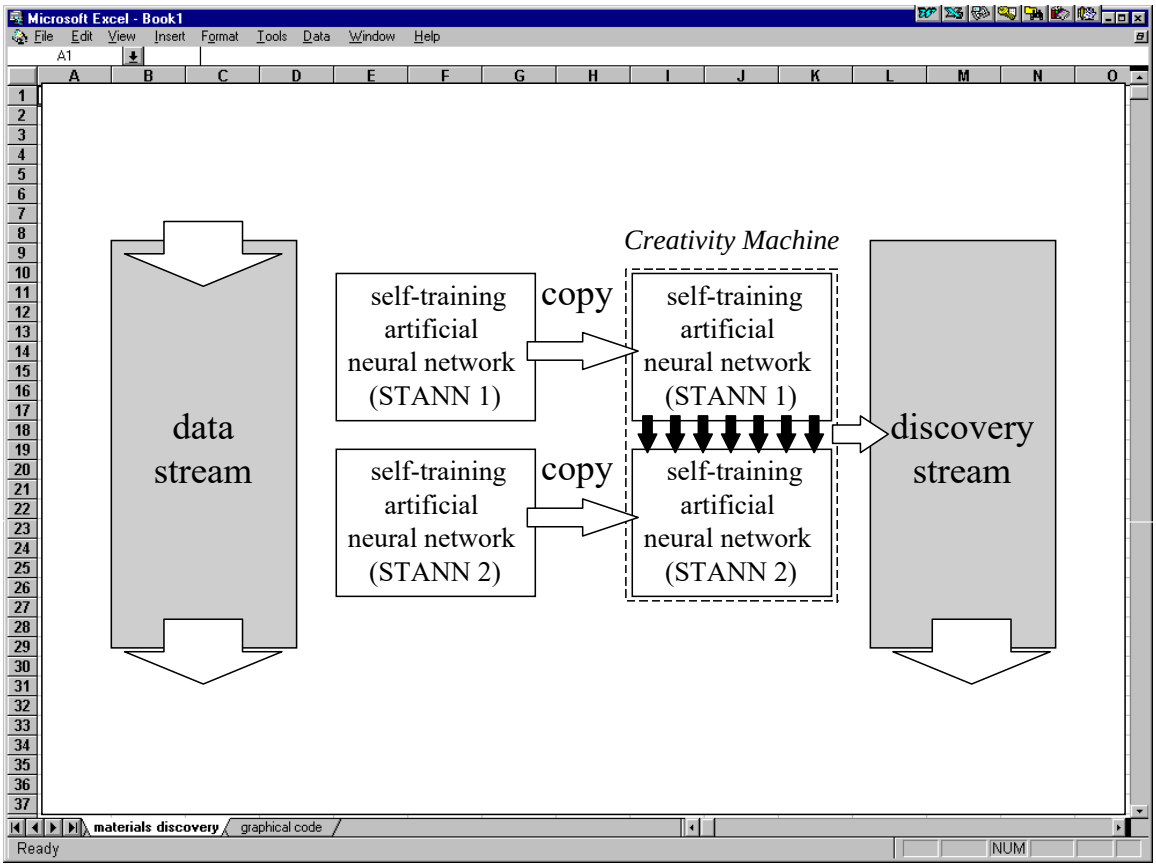


Figure 5. Autonomous discovery agent built into an Excel worksheet. A continuous stream of data arriving by DDE is watched by two self-training artificial neural network objects (STANNOs). These STANNOs are periodically copied and pasted into a Creativity Machine architecture. The Creativity Machine in turn produces a stream of discoveries and an accumulating discoveries database.

4. Creativity Machine Applications Achieved

At this writing several Creativity Machines are already operational, while myriad others are either under construction or in the planning stage. In the following section, I summarize some of the most recent functional machines and their spin-offs. In so doing, I attempt to express the wide range of topical areas this paradigm is applicable to.

Automobile Design - The overall objective of this Creativity Machine was to design an automobile to achieve any desired performance characteristics. Thus supplied with the objective of designing a car that achieves 35 miles per gallon, accelerates from 0 to 60 miles per hour in 7 seconds, and costs less than $30,000, the imagination engine, subjected to varying degrees of internal noise, will visit numerous examples of both exemplars and hybridized exemplars while a second net polices this stream of concept designs for one fulfilling the pre-established criteria. In this example, the imagination engine consisted of a feedforward net mapping 29 performance characteristics to 29 design specifications. We trained the policing net on the reverse mapping, relating design specifications to performance characteristics. In operation, we clamped the inputs of the first net at fixed values, while randomly applying perturbations to its weights. Self-consistent design specifications emerged at the first net's outputs while the second converted these outputs to anticipated performance values. The process continued until attaining the desired values. In more difficult design problems, a separate algorithm tracked those neurons that repeatedly contributed to the target designs. In this way it we were able to eliminate those processing units generally not carrying their share of the computational load.

Ultrahard Materials - Mostly as a reduction to practice of the Creativity Machine concept, I addressed the problem of predicting binary compounds with potentially ultrahard crystalline phases. In producing an imagination engine we

trained an auto-associative network on examples of binary compounds having the general formula A_xB_y. As a representation of each of the elements A and B, we used the ground state electron configurations respectively. For the stoichiometry factors x and y we used binary coded decimal representation. Provided with approximately 250 training exemplars of A_xB_y the network was able to correctly generalize the stoichiometries of any the arbitrary sets of elements A and B supplied to the net. With introduction of internal perturbation to this net, its outputs provided a stream of plausible, potential compounds. The second, policing net, trained on 250 exemplars to convert chemical formula to Knoop hardness, then sorted this resulting imagination stream for the hardest possible compounds.

Preliminary runs of this autonomous materials discovery machine have corroborated such recently proposed ultrahard materials as β-C_3N_4 (Liu & Cohen, 1989). Other ultrahard materials intimated by this Creativity Machine lie beyond this controversial gray theoretical area into purely speculative regimes, recommending, for example, totally new phases of hydrogen-deficient polymers of both beryllium and boron. Although no one has synthesized these materials, their candidacy as ultrahard seems likely owing to the high bond energy densities possible and the propensity for such materials to form highly interconnected network structures.

Currently, an improved version of this spreadsheet implemented Creativity Machine is under construction, incorporating an imagination engine trained upon thousands of examples of chemical compounds and relevant phase information gleaned from x-ray diffraction measurements. Once completed, this 'stoichiometry engine' will recommend totally new compounds and phases. Subsidiary networks will in turn map such hypothetical compounds to various technologically and commercially important physical and chemical properties such as hardness, electro-optical properties, and superconducting transition temperatures. In Figure 6 (A frequency histogram showing the prevalence of activated chemical memories and their generalizations) we see a preliminary imagination engine successfully predicting the oxides of iron. We consider most of these compounds to be 'discoveries' since only Fe_2O_3 has been shown to the network as a training exemplar. Also note within Figure 6 the growth of metallic iron along the plot's back wall, as oxygen concentration dwindles.

"Diamond-Like Nanocomposite" Process Optimization - In its first commercial application, I have built a Creativity Machine that automatically selects the optimal combination of process parameters within a coating chamber to achieve any desired thin film properties. Specifically, this machine recommends the correct reactor conditions to tailor films of a novel ultrahard material known as "Diamond-Like Nanocomposite" or "DYLYN™."[1] Thus given various customer requirements of hardness, elastic modulus, adhesion, stress, optical properties, a DYLYN Creativity Machine supplies the correct processing regime, consisting of 21 adjustable parameters. In building this Creativity Machine we have made optimal use of the graphical Excel interface to construct extensive cascades. By using a series of modular network models we have been able to relate control parameters to various diagnostic in situ characterizations, and these characterizations in turn to the various film properties. In this way, it is possible to observe and establish various process schema in elucidating the underlying physical and chemical models.

As an extension of this process optimization CM, we are now in the process of building a spreadsheet-based neurocontrol system that can monitor sensors and control actuators within the DYLYN reactor. Communicating with the reactor by DDE to its spreadsheet, we are building a CM that not only controls and maintains process set points, but also 'invents' various recovery paths to ensure any given set of targeted film properties.[2]

[1] A patented nanocomposite film produced by Advanced Refractory Technologies, Inc., Buffalo, N.Y.

[2] This work is funded under Air Force Contract AF-96-152, Automated Data Acquisition for In-Situ Material-Processing Modelling.

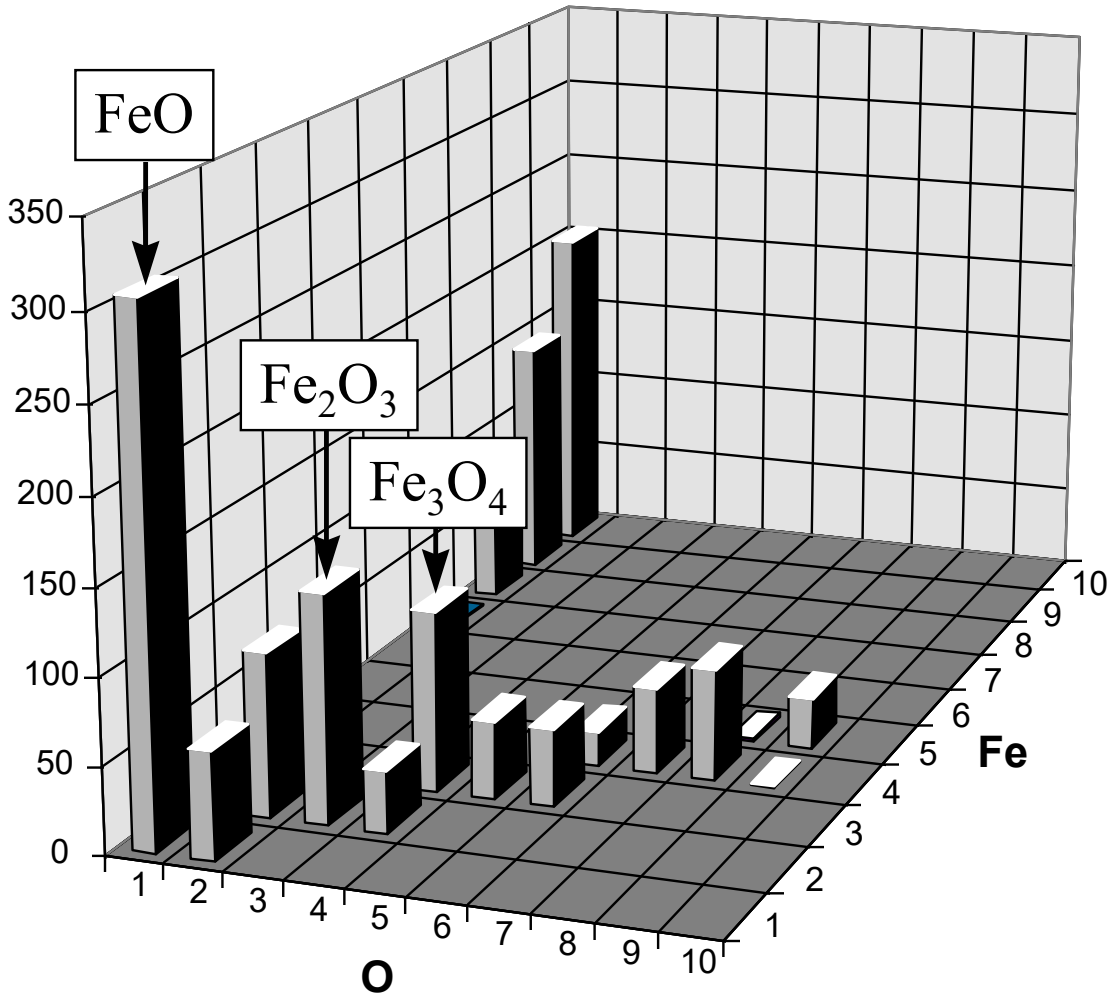


Figure 6. Concentrating only on two elements, the stoichiometry engine successfully predicts the oxides of iron. Only Fe_2O_3 has been shown to the network as a training exemplar.

Musical Composition - As a more subjective exercise, we have experimented with a Creativity Machine to generate novel musical melodies. The imagination engine, consisting of an auto-associative network, trained on 50 examples of "top ten" melodies or 'hooks' that have appeared over the last 30 years. I achieved the vector representation of these hooks by representing musical notes in both frequency (in hertz) and duration (in seconds). The policing net trained on data obtained from a group of panelists who listened to both accepted music, noise, and selected imagination engine output, offering numerical scores ranging between values of 0 to 10. As a reduction to practice, this Creativity Machine generated 11,000 novel musical hooks consisting of 10 successive notes. Recently completed is an expanded version of this CM that may generate 100 note melodies.

Targeted Medical Treatment - In its first medical application, we have constructed a Creativity Machine that recommends to an orthopedic surgeon the optimal pre-operative conditioning of a patient for hip implant surgery. In its first version, the machine computes a target patient weight and activity level that will ensure the highest possible success. This figure of merit consists of a lumped score of mobility, pain, and probability of infection. In a follow-on version of this machine, we are also building in a facility to advise the surgeon as to the preferable metallurgical composition for the hip implant.

More advanced medical applications, now in the planning stage, will devise the most effective order and degree of various treatments (e.g., chemical, radiological, and surgical) that stand to produce the most beneficial results against specific diseases. Building upon the fundamental CM paradigm, we may construct imagination engines that review a wide range of treatment scenarios while policing nets evaluate potential outcomes of those medical regimens. Taking

full advantage of the modularity of the spreadsheet implementation, we may add successively more policing nets to estimate efficacy, lethality, side-effects, etc. The machine would then seek an optimal efficacy while minimizing deleterious effects upon the patient.

Speech Recognition - Key to a successful speech recognition scheme is (1) the need for complex network cascades that generally emulate the multiple stages of auditory and linguistic processing in the human cortex, (2) the ability to differentiate spoken words from various sources of background noise, and (3) the ‘invention of significance’ to the captured speech pattern. In the first matter, we have found that the spreadsheet representation is most advantageous in linking the sundry networks that transform the raw amplitude-time signal to semantic representation (i.e., meaning). In the second issue, we have found that human speech possesses a distinctive fractal signature that is easily detectable within any time-evolving signal (Thaler, 1996c). We may therefore train one network within this complex cascade to recognize and hence gate the neural cascade to process the incoming voice signal. This feature enables the spreadsheet cascade to differentiate human from non-human sound sources.

We have achieved the third capacity of understanding by borrowing an important lesson from neurobiology, supplied largely by Freeman (1990). That is, within human perception, raw sensory input does not simply activate quiescent associative cascades corresponding to relevant concepts. Instead, mediated by internal chaos, various associative cascades compete to activate as a result of the incoming sensory signal. We have therefore been experimenting with whole ensembles of networks, bathed in internal noise, that effectively battle to activate once supplied a voiced word or phrase. A policing network watching this grouping then determines whether a winner has emerged from the competition and then inhibits those networks that have been less successful in attaching meaning. In this way, the Creativity Machine Paradigm invents significance to the spoken word.

6. Conclusions

The spreadsheet implemented Creativity Machine has proven to be an extremely powerful improvement to its original, purely algorithmic predecessors implemented largely within C-code. Because it creates a convenient graphical interface for the connectionist designer, we may prototype, test, and utilize extensive cascade structures corresponding to complex neural network models. What is more important, the spreadsheet representation has led to the development of a novel form of “unsupervised backpropagation,” allowing the spreadsheet implemented Creativity Machine to rapidly train on real-time data arriving by dynamic data exchange. Thus CMs may be adapt to evolving knowledge and conditions without interruption for retraining.

Spared the necessity to periodically update its associated networks, the Creativity Machine may interact with the external world through dynamic data exchange with external sensors and actuators. We thus pave the way to complex and sophisticated neurocontrol systems completely embedded within Microsoft Excel spreadsheets.

7. References

1. Freeman, W.J., and Skarda, C.S. (1990). “Representations: Who Needs Them?” J.L. McGaugh, et al., eds., *Brain Organization and Memory Cells, Systems and Circuits* (New York: Oxford University Press.
2. Liu, A.Y. & Cohen, M.L. (1989). *Science* **245**, 841.
3. Rowe, J. and Partridge, G. (1993). Creativity: A survey of AI approaches, *Artificial Intelligence Review* **7**, 43-70.
4. Thaler, S. L. (1993). 4-2-4 encoder death, *Proceedings of the World Congress on Neural Networks* 2, Portland, Oregon, 180-183.
5. Thaler, S. L. (1995). An Autonomous Discovery Machine for Ultrahard Materials, *Bulletin of the American Physical Society*, Program of the March Meeting, Series II, **40**(1), 592.
6. Thaler, S. L. (1995). Virtual input phenomena within the death of simple pattern associator, *Neural Networks*, **8**(1), 55-65.
7. Thaler, S.L. (1996a). Neural networks that create and discover, *PC AI*, May/June ‘96, 16.
8. Thaler, S.L. (1996b). A proposed symbolism for network-implemented discovery processes, To be published in the *Proceedings of the World Congress on Neural Networks 1996*.
9. Thaler, S.L. (1996c). Self-Training Artificial Neural Network Objects, To be published in the *Proceedings of the World Congress on Neural Networks 1996*.
10. Thaler, S.L. (1996d). Is neuronal chaos the source of stream of consciousness?, to be published in Proceedings of the World Congress on Neural Networks, WCNN’96, Lawrence Erlbaum & Associates.

11. Yam, P. (1995). As they lay dying...near the end, artificial neural networks become creative, *Scientific American*, 272(5), 24-25.

8. Acknowledgements

The author wishes to acknowledge the extremely helpful assistance of Mr. Daniel J. Redmond in the preparation of this paper and the oral presentation of its main themes.

"Databots"

S.L.Thaler
Dendrite Neurocomputing
12906 Autumn View Dr.
St. Louis, MO 63146-4331
(314) 576-1617
sthaler@ix.netcom.com

ABSTRACT

The ability to form extensive cell reference networks within typical spreadsheet applications lends itself well to emulating the central nervous systems of simple organisms. Exploiting this capability, we have successfully created artificial neurons, neural networks, and neural network cascades within Microsoft Excel worksheets. The resulting synthetic neurobiology not only forms the basis of adaptive artificial life, but also the foundation for automated knowledge base agents. These agents may (1) exercise their independent judgments in perusing databases, (2) choose their own perspective on or physically move through the data, (3) automatically learn hidden data patterns, and (4) cooperatively build compound cascade structures capable of autonomous discovery and invention. Provided that data passes through the spreadsheet environment, such "databots," acting alone or in concert, observe, generalize, and manipulate that influx of information. They may fulfill the whims of human researchers or, based upon their initiatives, exhaust a database of all potential discoveries. Advanced databot forms may autonomously assemble the needed algorithmic agents to infiltrate and explore larger databases external to their spreadsheet origin.

In this paper we review the principles underlying databot construction and function, drawing a one-to-one correspondence with the biological capabilities of advanced adaptive organisms. We show how databots achieve vision, attention, visual processing, novel feature detection, movement, learning, reproduction, and creativity using simple spreadsheet properties and methods. Drawing parallels with the notion of encapsulation in object-oriented programming, we also see how such databots achieve autonomy from both operator and operator-conceived algorithms. Related discussion then focuses on the databot's intentionality as self-originated decisions arise by allowing a supervised chaotic network to imagine various action scenarios through the so-called "Creativity Machine Paradigm."

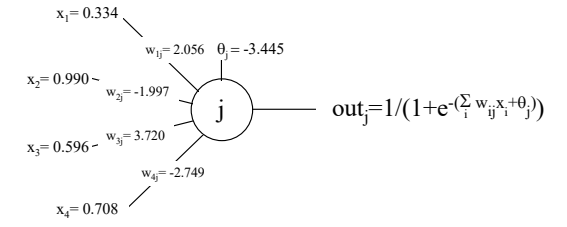
Offered tens of millions of cells within current spreadsheet applications and the possibility of increasing those numbers by several orders of magnitude in the future, we speculate on the potential and future applications of the databot principle.

1. The Synthetic Central Nervous System

A very powerful and flexible means of constructing and operating various parallel computing architectures is by means of spreadsheet techniques. We may implement Individual processing units, networks, and network cascades by the exclusive use of only cell references and resident spreadsheet functions. In Figure 1, for instance, we illustrate synaptic integration and output squashing using only relative cell references and a sigmoidal transfer function. The result is a neuron that is highly reproducible and transportable. As a consequence, we may join neurons into neural networks, and these networks into neural network cascades, all of which share the same mobility. We are at liberty to copy, cut, and paste these networks into arbitrarily complex architectures, freely interconnecting computational units and building any degree of recurrencies needed. If desired, we may even embed neural network models of actual biological neurons to serve as the individual processing units.

In the intelligent search of massive databases, we would like to create intelligent software agents that embody the human attributes of self-initiative, adaptivity, and creativity toward the objective of data management and discovery. To achieve this goal, the Excel spreadsheet application supplies a convenient neural network development environment within which we may prototype and refine the necessary neural cascades to emulate the needed human-like brain pathways. In essence, we seek to broadly simulate the human central nervous system (CNS) using the

spreadsheet using only spreadsheet resources. In correspondence to the biological situation, spreadsheet data now represents general sensory stimuli. Various spreadsheet-based associative networks may then parcel out these raw sensory inputs to subsequent and more specialized processing centers, thus emulating modularization within the brain. Subsequent spreadsheet networks may then 'invent' new courses of action within the context of the sensed environment and stored memories using the so-called "Creativity Machine Paradigm " or CM, described in more detail below. Finally, in affecting its environment, instead of using limbs and appendages, the distributed algorithm represented by the spreadsheet cascade may then direct a strictly algorithm code or 'slave macro' to carry out some action or movement solely originated within that cascade.



	A	B	C
1	0.334	2.056	1/(1+EXP(-(SUMPRODUCT(A1:A4,B1:B4)+B5)))
2	0.990	-1.997	
3	0.596	3.720	
4	0.708	-2.749	
5		-3.445	

Figure 1. Synaptic integration and subsequent output squashing (above) achieved within a spreadsheet implementation (below).

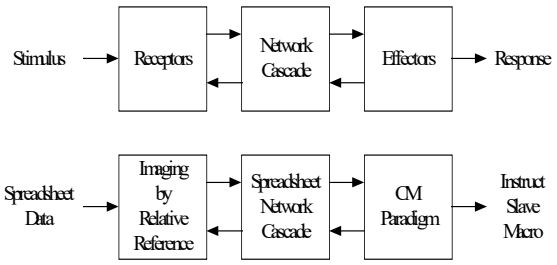


Figure 2. Block diagrams representing the nervous system (above) and its isomorphous spreadsheet implementation (below).

In this paper we discuss the achievement of all the processing stages depicted in Figure 2, within the Excel neural network development environment, and then describe several experiments within which the underlying principles have been reduced to practice. The result is a tool kit with which to synthesize these adaptive spreadsheet agents, we call "databots." These databots may then patrol and manipulate static data bases as well as monitor and learn from data streams supplied through dynamic data exchange (DDE) to Excel.

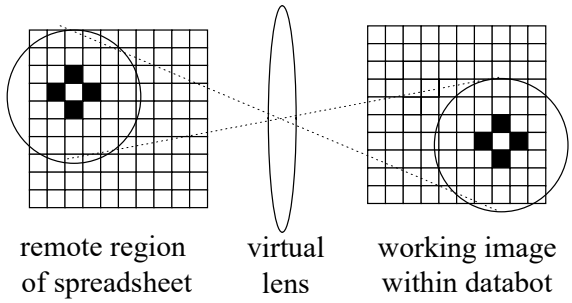


Figure 3. Formation of a working image by relative cell references in spreadsheet application.

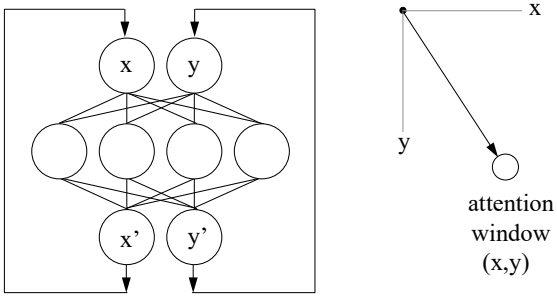


Figure 4. Use of recurrent autoassociative map to emulate population coding to guide foveation.

2. Databot Receptors

Sensory inputs to the databot are primarily 'visual,' based on the notion of locating and interpreting data resident within a spreadsheet application. In so doing, the databot must produce some working replica of its data environment prior to deeper visual processing (Figure 3). In analogy to the formation of a retinal image in neurobiology, the databot forms at least one working replica of its target data by relative cell reference. In some cases, this image may be as simple as single numeric values, as in supplying the input values to a feedthrough network. Within other requirements,

the working image formed may represent a larger receptive field containing a cluster or grouping of spreadsheet objects.

Central to databot function, this synthetic organism must be able to scan the environment within its vicinity. One such scheme that I have successfully used to facilitate such a visual scanning process involves the emulation of the neurobiological strategies known as *population coding* and *vector averaging* (Churchland & Sejnowski, 1992). In such processes an external sensory organ follows the 'center of gravity' of activation among a colony of highly interconnected neurons. In the brain, for instance, the superior colliculus is a type of "where to foveate next" map for the eyeballs. The final directions of movement and resting position depend upon all the neurons in the colony contributing activity to various degrees. I have used recurrent associative networks, as exemplified in Figure 4, to implement this type of chaotic scan. By constantly recirculating the outputs x'-y' back to the inputs x-y, simultaneously introducing internal perturbations to the network's connection weights, the attention window centered at x-y wanders the spreadsheet in search of specific spreadsheet items. Apart from its observed speed, the major advantage of this search technique is that activation from networks viewing items within this attention window may modulate and hence diminish the rms noise amplitude applied to network weights. Because of this feedback, the point of foveation tends to gravitate toward and ultimately center on objects of interest (Figures 5 & 6).

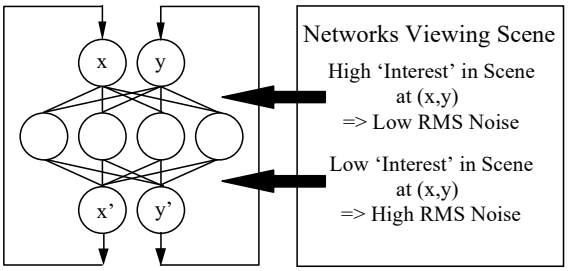


Figure 5. Viewing networks provide feedback to population coding net by modulation of noise amplitude applied to its weights.

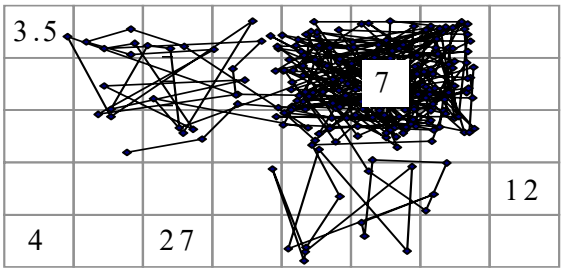


Figure 6. Databot foveation trail gravitates toward sought data. Here, the viewing net has been trained to activate upon detection of prime integers.

Often the databot must be able to quickly gravitate toward large clusters of information, moving either in the plane of the worksheets or in effectively 3 dimensions, through successive pages of a workbook. To exemplify how to achieve this task, we may imagine databot retinas segmented to act as quadrant detectors. Concentrations of data in the x, y, -x, and -y directions register instantaneously to provide an information gradient to the organism. The vector sum of data located in each of these quadrants provides a movement trajectory. Therefore, in seeking specific information, the databot will tend to exhaust the larger data clusters before moving on to the more scattered information.

3. The Internal Databot Network Cascade

The spreadsheet cascades that process the databot's raw sensory input (i.e., the central network block of Figure 2) handle several specific tasks including (1) location, (2) identification, (3) learning (4) novel feature detection, (5) attention windowing, and (6) novel movement and activity planning. The first three processing categories represent instances of information processing, while the latter two closely related tasks embody information generation. Here we briefly touch upon each of these levels of internal databot function, pointing out the general principles and problems surmounted.

3.1 Location and Identification - The problems of location and identification have been best handled in a manner reminiscent of human visual processing, generally parceling out the preliminary retinotopic maps at the visual cortex into the well-known dorsal and temporal channels. These two channels represent a rough dichotomy of the problem of 'where' and 'what' in visual processing (Kosslyn & Koenig, 1992) and solution to *the problem of stimulus equivalence across retinal translation* (Gross & Mishkin, 1977; Rueckel, Cave, and Kosslyn, 1989). In Figurea 7 and 8, we show how using two separately trained spreadsheet networks we may attain increased classification accuracy. In general, we resort to similar modularization of tasks within the databot's synthetic central nervous system.

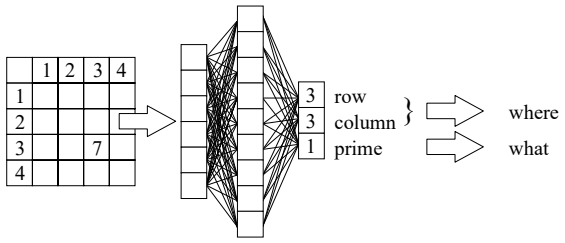


Figure 7. A databot network both locates and identifies a prime number within a spreadsheet. Mixing both chores within the same net tends to degrade accuracy.

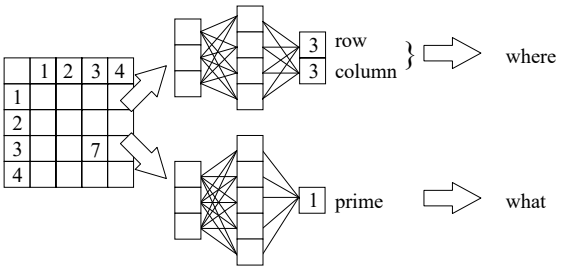


Figure 8. By division of labor between two neural nets we more accurately locate and identify spreadsheet data.

3.2 Learning - Anticipating that the chief function of the databot will be to glean various trends and patterns from massive databases, it will be necessary to incorporate some form of learning paradigm within the databot's synthetic CNS. In the spirit of supplying complete autonomy to these spreadsheet agents, we may utilize a novel training technique in which we embed the backpropagation paradigm as a distributed algorithm within a spreadsheet network, connected in turn to an untrained spreadsheet network (Thaler, 1996). It is the backpropagation net that essentially watches both the inputs and output errors of the first and then iteratively adapts connection weights to capture any patterns within the observed data. We accomplish the critical step of calculating the local partial derivatives of neuron activations with respect to net by submitting infinitesimally incremented inputs to a replica network with the backpropagation supervisor net observing the change in neuron activations. Viewing the two interconnected networks as a whole, we may consider it to be a "self-training artificial neural network object" or STANNO, that need only be physically juxtaposed with the training data to spontaneously learn.

3.3 Novel Feature Detection - To sense data generally not falling within the pattern of information cumulatively observed by the databot, we may make use of autoassociative nets. Specifically, we find that after training an autoassociative map within any conceptual space of vectors $\mathbf{v}_i$, that the application of any vector $\mathbf{v}_e$ not belonging to that space to the inputs of that autoassociative map produces an output vector $\mathbf{v}_e$' differing from $\mathbf{v}_e$. In a sense, this phenomenon is analogous to linear matrix theory wherein only the eigenvectors of the transformation remain undistorted following matrix multiplication. Representing the autoassociative net as the nonlinear operator **A**, the unitary transformation as **1**, and the differential vector between network input and output as $\boldsymbol{\delta}$,

$$(\mathbf{A} - \mathbf{1})\, \mathbf{v}_e = \boldsymbol{\delta}. \qquad (1)$$

Hence, the more unique the vector $\mathbf{v}_e$ to the majority of data cumulatively 'seen' by the databot, the greater will be the magnitude of the vector $\boldsymbol{\delta}$. We may therefore use $\boldsymbol{\delta}$ as an indicator of pathological or spurious data. Alternatively we may use this difference vector to gate novel information into a self-trainer, sparing it from exposure to redundant training exemplars.

3.4 Attention Windowing - Combining the above ideas of population coding and autoassociative novelty detection, a databot may peruse its spreadsheet landscape in search of unique data features, in a manner reminiscent of eye saccades in animals. Using the autoassociative novel detection scheme described above, the databot may then dwell on the anomalous feature. It may then elect to self-train on the new item or to perform any number of possible spreadsheet manipulations upon it.

4. Databot Effectors - Using spreadsheet neural networks typical of those incorporated into databot designs, we may create specialized cascades containing tandem pairs of nets in the so-called "Creativity Machine" architecture (Thaler, 1996). Succinctly stated, one of these networks sustains progressively increasing levels of perturbation to its weights and biases, as it activates through a series of states reminiscent first of its training exemplars, then ultimately evolving into generalizations of these exemplars at higher levels of internal disruption. Many of these latter network activations represent confabulations of the unperturbed network's memories, or new ideas, if you will. Many of these new ideas

may be of use, being judged as worthy by the second associative network watching the activations of the first, perturbed net. Using this paradigm, we may invent or discover within any arbitrary conceptual space, provided that the requisite database of training exemplars exists.

As an extension and generalization of this creative process, we imagine that the most mundane of databot movement or action planning is implementable out by the Creativity Machine paradigm. That is, the perturbed net may represent the memory of the range of movement and past motion scenarios of the databot cascade (e.g., its allowed dimensions of travel, steric hindrances to motion due to spreadsheet obstacles, etc.). When internally perturbed, this net visits the entire phase space of potential movement scenarios. The second net may then choose some imagined movement that contributes to some predetermined goal. Constraining the first net by the introduction of auxiliary inputs may assist in further constraining the movement planning search and hence accelerating the databot's internal decision process.

We note at this stage that the databot is autonomous in the generation of its own decisions. That is, beyond any preliminary network training, the databot acts in isolation from either operator or peripheral computer codes. The only access to the databot's internal function is by spreadsheet data through its sensory cells. These cells fulfill a role similar to that of the public interface, so familiar within the context of object-oriented programming [i]. The major difference is that the databot represents a distributed algorithm rather than a sequential algorithm such as C or C++.

5. Response - In retrospect, we have built a synthetic nervous system, using the many analogies between a spreadsheet cell and an actual biological neuron. The result is a distributed algorithm, embedded within a spreadsheet application and capable of originating its own movement and actions, by imagining such scenarios in advance (Figure 8). Instead of the typical locomotive and manipulative paraphernalia utilized by biological organisms, we may supply the databot with recourse to various algorithmic routines (broadly segregated into movement and actions in Figure 9).

In many respects, the databot incorporates basic principles behind object oriented programming in that it contains internal data and functions hidden from peripheral algorithmic code (i.e., the underlying spreadsheet machine code and slave macros), from other databots, and the operator. The only information to penetrate this *encapsulation* is sensory data from the spreadsheet environment, with the sensory apparatus being tantamount to the *public interface* of a class object. Decisions to move, reproduce, or perform manipulation upon data is therefore strictly an internal decision, granting the databot (what many would agree) status as an autonomous agent.

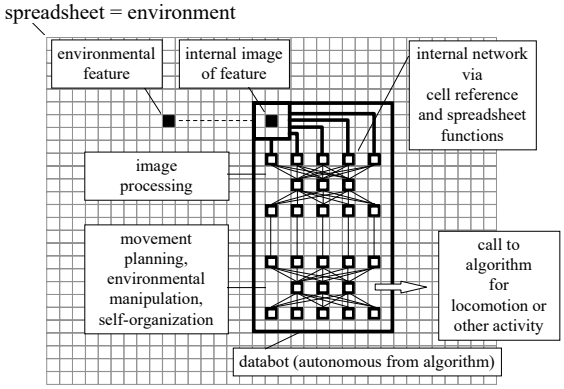


Figure 8. Summary of overall Databot function and the use of a 'slave algorithm' to influence its spreadsheet environment.

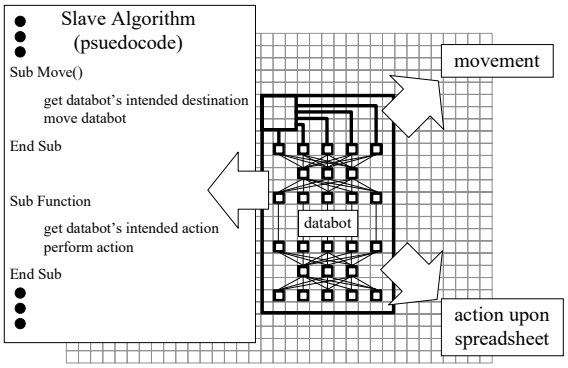


Figure 9. Response of the databot originates within the databot's synthetic CNS. It may now interact with its environment by calls to appropriate macros or subroutines.

In effecting any given movement imagined by the databot, we may enlist any number of simple spreadsheet facilities. For instance, we may achieve incremental or smooth databot motion by successive cut and paste commands. Because the databot's CNS construction uses relative cell referencing, the synthetic creature relocates itself as an intact, functional unit at its updated position. The net effect is that of continuous translation. Similarly, using copy and paste commands, databots may clone, perhaps in response to an increase in data volume and the need for multiple databots to take myriad perspectives and training upon the arriving information stream.

6. Applications - Each of these spreadsheet schemes emulates various aspects of CNS function. We may packaged them as individual neural network modules and then link them together to perform a wide variety of functioning databots. They would then serve as specialized agents to autonomously perform varied spreadsheet database functions. Below we enumerate just a few potential applications of these virtual organisms utilizing the repertoire of capabilities described above and summarized in Table 1. We mat link together any combination of the modules together within the databot's synthetic CNS.

6.1 Database Management - Databots may exercise their initiative in organizing and manipulating information within spreadsheet databases using many of the principles outlined above. One example already implemented is the detection and isolation of experimental data generally not falling within preexisting patterns. It is a matter of context and databot objectives whether to treat such data as pathological or novel. Typically, such databots have used the autoassociative principles already described to identify anomalous data vectors, along with either the ability for locomotion among the data or, for static databots, a foveation mechanism to scan spreadsheet information. The databot then decides autonomously whether to mark that data for the benefit of the operator or to cut and paste such pathologies into an archival location.

One facet of database management already reduced to practice in preliminary experiments is the use of databots to search for specific items embedded within a spreadsheet workbook. In these trials, databots have been successful in retrieving geometric objects tucked away within a clutter of distracting data. Such work is preliminary to employing databots as "store room clerks" that seek various items using *content addressing* schemes. Therefore it is possible to show a trainbot numerous examples of pentagons, for instance, and have it locate and transport similar shapes to a predetermined collection area within the workbook. In some applications, databots have been equipped with special 'enzyme faces' conformal with the objects to be collected. Upon locating the sought item, the databot may group itself with the object and then return to the collection point where it commands a slave algorithm to ungroup it from its baggage. Generalizing this principle we may easily envision teams of inventory databots that may provided with relatively ambiguous search criteria, ferretting out obscure objects on the basis of a wide variety of criteria.

6.2 Database Discovery - As multiple databots, trained on distinct perspectives to the data update themselves in terms of their prediction accuracy, they may autonomously elect to copy and paste themselves into complex Creativity Machine cascades to both create and invent within the specific problem space. That is, these virtual organisms may coordinate into larger collective structures for the purpose of autonomous discovery.

Another important aspect of database discovery deals with the elucidation of various *schema* stored within the connection weights of an ANN. That is, during training, a network's connection traces grow in magnitude and sign to reflect the logic or models underlying the conceptual space under scrutiny. To reveal such schema, databots may successively remove various connection weights from the trained network, on a trial basis, assessing each weight's importance in maintaining the fidelity of mapping. Should we judge any weights superfluous to that mapping, we remove them in a process described as *skeletonization* (McMillan, Mozer & Smolensky), as exemplified in Figure 10. Teams of databots, each trained to recognize neural traces corresponding to various logical, comparative, and arithmetical operations, may overlay corresponding graphical symbols onto the neural network (e.g., AND, OR, and NOT gates) to form a semantic or logical network. In the case of Figure 10, we form a schematic network.

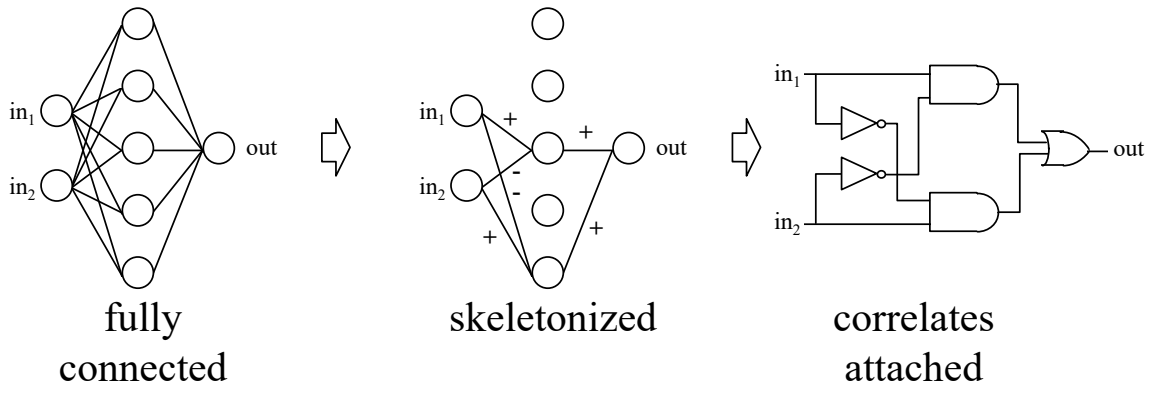


Figure 10. Databots skeletonize fulley connected XOR net, attach correlate symbols in the form of AND, OR, and NOT gates, elucidating function of original net.

6.3 Robotic Prototyping - With connectionism playing an increased role in cybernetics research, it would be ideal to possess a breadboarding utility to test various robotic principles. That is, before the creation of any hardware such as ROM, we may readily prototype various autonomous movement, vision, and speech facilities. We may then evaluate the function of autonomous agents and devices prior to committing designs to production.

6.4 Infiltration of External Databases - Until now, I have discussed scenarios in which data passes through the spreadsheet application by for instance DDE, where we may apply various data processing stages. It is also possible to allow databots to assemble algorithmic code, which may in turn be used to infiltrate larger, external data stores. In preliminary experiments, databots have successfully constructed code segments by cutting and pasting Visual Basic program steps into usable macros. That is, the databot chooses ASCII commands within the spreadsheet, and then pastes these steps into the VBA macro environment, in the appropriate program order.

6.5 Artificial Life Laboratory - Given this degree of independence, it should be interesting to initiate colonies of these virtual creatures, allowing facilities for mutation and natural selection. The interesting new dimension to this GA type theme is that what is mutating are these creatures' central nervous systems, hopefully evolving into more sophisticated synthetic organisms more adept at data mining and discovery. Perhaps using such techniques we may breed successively smarter intelligent agents.

6.6 Dynamic, Adaptive, Intelligent Agent in a Spreadsheet -Databots may collectively self-organize into a colonial intelligence capable of human level cognition. Considering the potentially hundreds of millions of spreadsheet neurons possible and the databot's inherent self-training facility, it may be possible for databots to autonomously assemble first individual brain modules, link such modules, and then collectively train the overall brain structure via the self-training paradigm. The resulting agent would then be capable of conversation, reasoning, and seminal thought.

Table 1. A sampling of databot functions.

Databot Function	Spreadsheet Implementation
Vision.....	Relative cell references produce duplicate image of data.
Attention.....	Creativity Machine invents an attention window on the data.
Visual Processing.....	Image of data is handed to databot's neural network cascade.
Autonomy.....	Vision and visual processing are implemented completely in neural networks using only resident spreadsheet references and functions. Thereby encapsulated from any macros or human intervention, these neural networks then command a slave algorithm, composed of VBA macros, to effect complex actions such as motion.
Movement.....	Slave algorithm cuts entire databot consisting of relative cell references and functions and places them in a location predetermined by the databot, using a specialized Creativity Machine.
Learning.....	Databot uses one neural network to train another untrained network.
Novel Data Recognition.....	Databot uses autoassociative net to filter data prior to any learning.
Creativity.....	Internal Creativity Machines.
Discovery.....	Databot uses dynamic skeletonization of its trained neural networks to understand interplay of factors within its spreadsheet environment. Teams of databots are capable of replacing the revealed neural traces with semantic representations of them.

Reproduction.....	Databots may fission by appeal to a slave algorithm which copies and pastes them. Sexual reproduction or mutation may occur by similar databot request to the slave algorithm, allowing for natural selection processes.
Data Manipulation.....	Databots may appeal to the slave algorithm to delete, copy, cut, and paste data. The databot will determine where to position any displaced data through its internal classification schemes.
Data Binding.....	In addition to willing a slave algorithm to group data, the databot may be equipped with an 'enzyme' face which selectively attaches the sought data.

6.7 Universal, Adaptive Control Systems - The ultimate spreadsheet based neurocontroller would involve a colony of databots that can flexibly adapt to any number and nature of DDE inputs. In this context, databots could engineer the necessary sensory channels and autonomously bridge inputs to control outputs by the construction of the required intermediate cascades.

7. Conclusions

It has been more than a decade since Valentino Breitenberg's (1984) classic work Vehicles, Experiments in Synthetic Psychology, involving a series of thought provoking gedanken experiments with connectionist contraptions that mimicked all aspects of human and animal behavior. In the interim Edelman and coworkers (see, for instance Franklin, 1995) have produced a series of computer models known as *selective recognition automata* that can categorize, as we do, within largely novel environments. I believe that the databot is a useful outgrowth and refinement of these seminal works, culminating in an independent synthetic organism that may prove especially important, not only in the discovery of hidden patterns within massive databases, but for the prototyping of artificial, adaptive life.

Stopping to consider that present day spreadsheet applications may contain 10 or even hundreds of millions of spreadsheet cells, and that any focused cognitive task may invoke similar numbers of biological neurons, it is my opinion that we are at the threshold of emulating human level intelligence. Provided the facility to self-train, all we need add is perceptual input.

7. Bibliography

1. Berry, T.J.(1992). *C++ Programming* (Carmel: Prentice Hall).
2. Braitenberg, V. (1984). *Experiments in Synthetic Psychology* (Cambridge: MIT).
3. Churchland, P.S. & T.J. Sejnowski (1992). *The Computational Brain (*Cambridge: MIT) 234-237.
4. Franklin, S. (1995). *Artificial minds* (Cambridge: MIT).
5. Gross, C.G. & Mishkin, M. (1977). The neural basis of stimulus equivalence across retinal translation. In S. Harnad, R. Doty, J. Jaynes, L. Goldstein, & G. Krauthamer (Eds.) , *Lateralization in the nervous system* (New York: Academic Press).
6. Kosslyn, S.M. & Koenig, O. (1992). *Wet Mind: The New Cognitive Neuroscience*, (New York: Free Press), 60-65.
7. McMillan, C., Mozer, M.C., and Smolensky, P. (1991). *Proceedings of IJCNN-International Joint Conference on Neural Networks IJCNN'91 Seattle*, Publ. by IEEE, IEEE Service Center, Piscataway, N.J., 83-88.
8. Rueckel, J.G., Cave, K.R., & Kosslyn, S.M. (1989). Why are "what" and "where" processed by separate cortical systems? A computational investigation. *Journal of Cognitive Neuroscience*, 1, 171-186.
9. Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition. Vol. 1* (eds Rumelhart, D.E. & McClelland, J.L.) (Cambridge: MIT)
10. Thaler, S.L. (1996). Neural networks that create and discover, *PC AI*, May/June, 1996, 16-21
11. Thaler, S.L. (1996). Principles and applications of the self-training artificial neural network, *Proceedings of the Advanced Distributed Parallel Computing Symposium, ADPC'96, Dayton Ohio.*

Principles and Applications of the Self-Training Artificial Neural Network

S.L.Thaler
Dendrite Neurocomputing
12906 Autumn View Dr.
St. Louis, MO 63146-4331
(314) 576-1617
sthaler@ix.netcom.com

ABSTRACT

Traditional supervised neural network trainers have deviated little from the fundamental backpropagation algorithm popularized in 1986 by Rumelhart, Hinton, and Williams. Typically, the training process begins with the collection of a fixed database of input and output vectors. The operator then adjusts additional parameters such as network architecture, learning rate, momentum, and annealing noise, based upon their past experience in network training. Optimizing the network's generalization capacity usually involves either experiments with various hidden layer architectures or similar automated investigations using genetic algorithms. Along with these often complex procedural issues, usable networks generally lack flexibility, beginning at the level of the individual processing unit. Normally, the user finds him or herself confined to a limited range of unit activation functions, usually including linear, linear threshold, and sigmoidal analytical forms whose partial derivatives with respect to net input are definable through a similar continuous, analytical expression. Generally, the demand is for a more flexible and user friendly system that will not only lessen the technical confusion for non-connectionist end-users, but also create expanded utility for those demanding more architectural freedom and adaptability in their artificial neural networks or ANNs.

By implementing an ANN within a spreadsheet environment, using only relative cell references and supplied functions, we create a quasi-parallel computational scheme quite distinct from past algorithmically based neural network simulations and addressing the above-mentioned needs. By allowing one such spreadsheet network to serve as training supervisor to another, we create an adaptive neural network cascade that need only be physically juxtaposed with the database to learn any of its contained patterns. Viewed as a whole, the combination of untrained network and its supervising counterpart represent a self-training artificial neural network, learning by parallel processing and without the assistance of algorithmic code. Providing some means to stream data past this self-training artificial neural network, or "STANN" (patent-pending), this agent "sees" exemplars and spontaneously self-organizes to generalize the represented knowledge domain. Such a network implementation is well suited for monitoring and learning from data streams supplied to the Excel environment, either from external devices, human input, or algorithmic codes. Furthermore, because such networks function in a parallel fashion, we may train multiple STANNs simultaneously, allowing us to gain multiple perspectives on the influx of data.

Because such self-trainers numerically calculate the needed partial derivatives, the functional form of activation need not be simple. Instead, each processing unit's activation function may be arbitrarily complex, perhaps taking the form of an entire neural network. We thereby achieve the capacity to create self-training neural network cascades, with each component network emulating the behavior of any given analog or digital device. Training of such compound networks would allow any combination of electrical, mechanical, or optical devices to self-organize and interconnect to achieve any desired input-output relation, in turn allowing us to rapidly prototype and invent a wide variety of hardware devices.

Herein we elaborate on the basic operation of the self-training artificial neural network and then expand on its potential applications and derivative concepts.

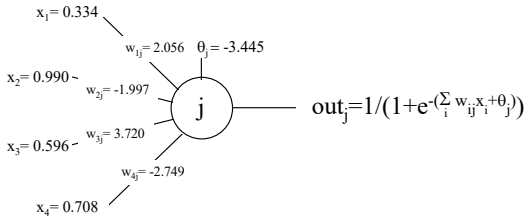
1. Introduction - We typically think of neural network simulations as the sequential evaluation of activation states of neurons within a network, using some algorithmic language such as C or C++. Within such schemes, individual activation levels are only momentarily visible and accessible, as when the governing algorithm evaluates the sigmoidal excitation of any given neuron (Figure 1). Except for its fleeting appearance during program execution, a neuron's excitation becomes quickly obscured by its redistribution among downstream processing elements.

Exploiting the many analogies between biological neurons and cells within a spreadsheet, we may evaluate the state of any given network processing unit through relative references and resident spreadsheet functions (Figure 2). By referencing the outputs of such spreadsheet neurons to the inputs of other similarly implemented neurons, we may create whole networks or network cascades. Unlike the algorithmic network simulation, all neuron activations are simultaneously visible and randomly accessible within this spreadsheet implementation. More like a network of virtual, analog devices, we consider this simulation to be a distributed algorithm, with all neurons equilibrating with one another following each wave of spreadsheet renewal.

```
void feedforward(float *input, float *output)
{
        ●
        ●
        ●
        for(j=0; j<nodes[lay]; ++j)
                act[j]= 1.0/(1.0+exp(-net[j]));
        ●
        ●
        ●

}
```

Figure 1. Neuron activation within a given network layer is evaluated in C-code.



	A	B	C
1	0.334	2.056	1/(1+EXP(-(SUMPRODUCT(A1:A4,B1:B4)+B5)))
2	0.990	-1.997	
3	0.596	3.720	
4	0.708	-2.749	
5		-3.445	

Figure 2. A representative neuron implemented in Microsoft Excel.

As a further benefit of the spreadsheet implementation, the user now has a convenient graphical interface for constructing and experimenting with ANNs. For instance, we need only build a template neuron once, using simple copy and paste commands to build and connect whole networks from a prototypical processing unit. We may repeat these procedures on larger scales to move networks into position and link them into larger cascade structures. The compound neural networks that result are transparent in operation and easily accessible for modification and repair. No longer confined to simple feedforward architectures, we may readily introduce recurrencies and all manner of neural network paradigms, including IAC, Boltzmann Machine, Harmonium, Hopfield nets, and self-organizing maps.

Originally intended to serve as an expedient means to produce spreadsheet network cascades we have discovered many additional advantages to this graphical network implementation. Probably the most important development to emerge from this scheme is the patent-pending concept of a 'self-training artificial neural network' or "STANN." Consisting of one network training another, we may view the STANN as a spin-off of the so-called "Creativity Machine" paradigm (Yam, 1995). Such Creativity Machines consist of a chaos-driven network, trained within a given knowledge domain, and supervised by a second network. As the level of internal perturbation is slowly ramped up in the former net, the second supervisory net monitors its output, alert to any emerging vectorialized ideas that fulfill some predetermined need or goal. Notable exercises of this paradigm have included the generation of myriad, esthetically pleasing musical hooks (Thaler, 1994) as well as the discovery of new ultrahard materials (Thaler, 1995). Because we may rapidly train both component networks by example and then control the chaotic network's "egress" from the chosen knowledge domain (by progressive weight perturbation), we may rapidly prototype and refine these systems. The technique offers the significant advantage of reduced development time over past rule- and model-based attempts at discovery and creativity (Rowe & Partridge,1993).

Viewed in this context, the STANN is comparable to an 'inverse', spreadsheet-implemented Creativity Machine. Rather than subjecting the connection weights of the first network to randomizing influences, we apply weight updates obtained from the supervising network. These weight adjustments ultimately correct the first net's initially randomized connection weights to achieve the desired mapping (Figure 3). Important to emphasize is that the supervising network calculates all weight updates from the observed errors, using a distributed algorithm built upon the traditional backpropagation paradigm. (There are no sequential algorithmic routines corresponding to the partial derivatives, error terms, and updates of the prevalent network trainers.) The result is that such a spreadsheet-implemented network cascade quite literally need only be shown data to train. For each exemplar presented to the STANN, a self-correcting wave propagates through the composite network, beginning with a spreadsheet calculation of network error and terminating at the network's updated connection weights. All the needed operations reside within the cell references and resident spreadsheet functions of the supervising network.

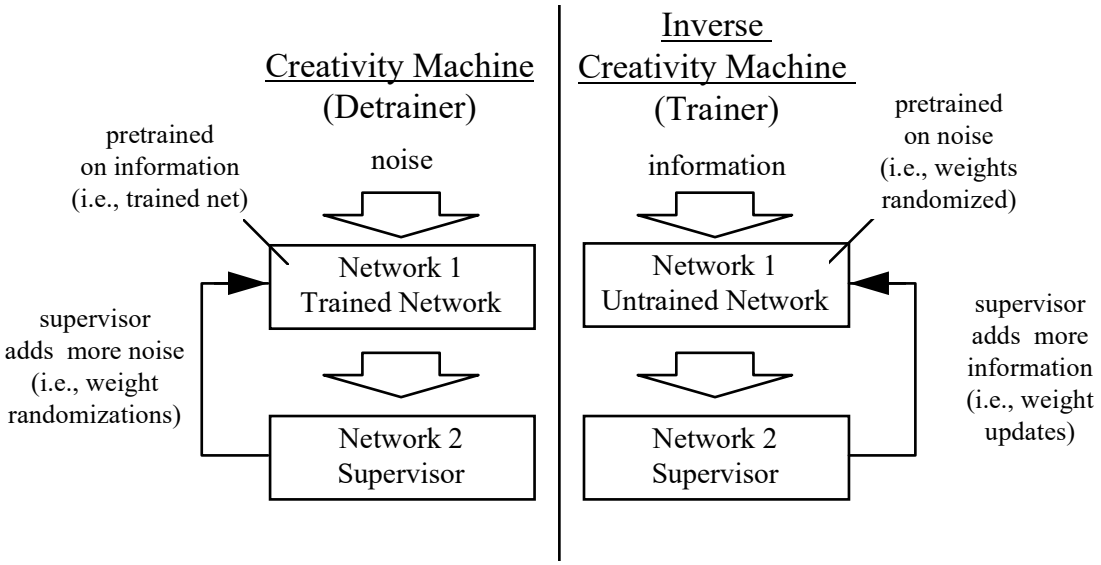


Figure 3. Considering noise and information to be complementary, the inverse of a Creativity Machine is a self-training neural network.

Viewing the untrained net and its supervisor as a single, composite net, adapting to data external to itself, the boundary between supervised and unsupervised learning becomes diffused. After all, except for the repeated application of the intended input-output to the net, there is no supervisory algorithm per se, simply a single composite network internally adapting to its external (spreadsheet) environment. The distinction between 'supervised' and 'unsupervised' totally dissipates when we purposely use an autoassociative network, and consider training on exemplars containing identical input and output vectors. Training then consists of applying these vectors to both inputs and outputs simultaneously while hidden layer neurons connect themselves to attain some classification scheme. The resulting process bears a close resemblance to the unsupervised methods characteristic of Kohonen (1982) self-organizing maps.

2. STANN Structure and Operation - Using only relative references and resident spreadsheet function, I have built several prototype STANNs. Whereas, we have implemented such STANNs in myriad ways using these underlying principles, their general construction (Figure 4) usually involves clearly differentiable modules corresponding to the individual stages of the backpropagation paradigm (Rumelhart, Hinton & Williams, 1986). All of these modules communicate through cell reference thereby binding them as a unit, embedded alongside the spreadsheet data. The distributed algorithm then estimates the partial derivative of each neuron's activation with respect to its net input by adding some small increment, δ, to the input vector's components and submitting them to a replica network contained in module 2. The STANN then numerically calculates the derivative in module 3 for all neurons by noting the changes in both its activation and its net input between modules 1 and 2. Modules 4 and 5 handle the backpropagation of error, within similar cell reference networks, with the resulting corrections simultaneously appended to all weights and biases in both the parent and replica networks in modules 1 and 2.

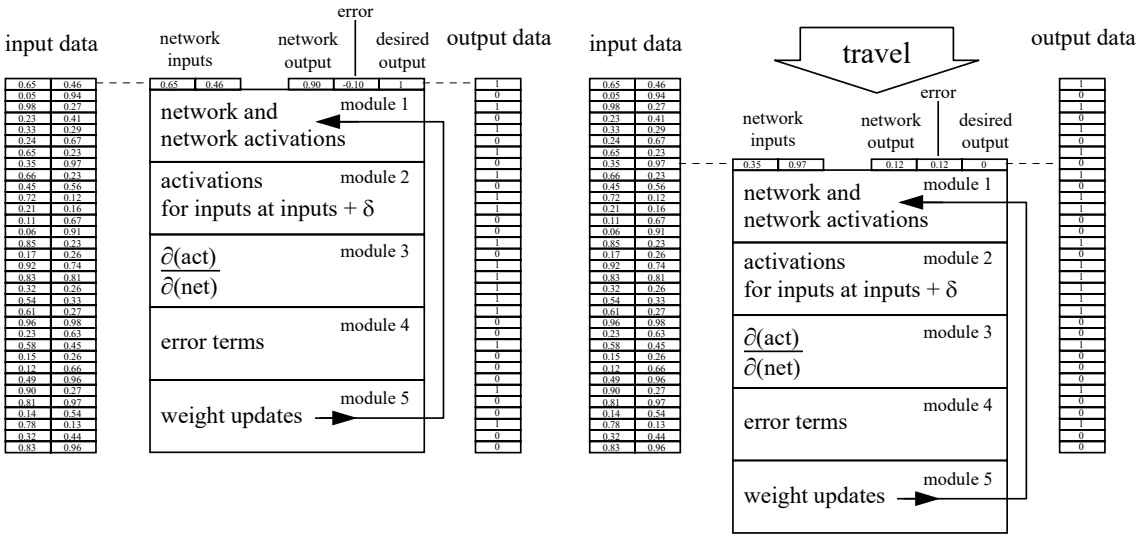


Figure 4. Module structure of a simple STANN (left) consisting of two inputs and one output, immersed within a database. The STANN mobilizes itself by automated cut and paste commands to move through the data to learn any contained patterns.

Important to note is that the STANN "sees" its training data by relative referencing. Thus any input data to the left of the STANN's network input cells appears duplicated within those cells. Similarly any output data to the STANN's right appears reproduced simultaneously in the designated output cell. Viewing these "interface" cells between the external spreadsheet "environment" and the STANN's internal network as metaphorical "retinas," we see that data may appear as a training progression to the STANN if data and STANN move relative to one another.

We achieve such relative movement between the STANN and the data by either (1) movement of the data or (2) movement of the STANN. In the former mode, data streams through the Excel columns by dynamic data exchange as the STANN observes a fixed group of spreadsheet cells. Alternately, STANN motion may occur by allowing the STANN to command an algorithm (i.e., Visual Basic macros) that performs cyclic cut and paste commands to the totally self-contained network cascade, as depicted in Figure 4.

The former training technique may find immense utility in the real-time patrol of data streams from sensors in various hardware or laboratory systems. Such STANNs learning in real time, may provide appropriate outputs to control various system actuators. In another application, separate VBA macros running algorithmic models may generate streams of model exemplars to a STANN to convert various rule- or model-based algorithms into their corresponding neural network models. Extensions of this concept may vastly facilitate the conversion of expert systems to their connectionist counterparts.

In the latter training technique, STANNs may move and train within static or periodically updated spreadsheet databases. There, the added dwell time allows the STANN to make many passes through the data (e.g., training epochs) between database updates. This concept finds added utility in advanced STANNs that are able to move in two or three spreadsheet dimensions, choose their own perspective on the database, and elect their own trajectory through the spreadsheet workbook. Such mobile STANNs or "databots" now can exhaustively search through either fixed or dynamic databases in search of subtle patterns of interest or unexpected discoveries.

3. Dynamic Data Exchange to STANNs - One exciting application of the self-training artificial neural network object is in the area of real time training of sensor data fed directly to spreadsheet environments by dynamic data exchange or 'DDE.' That is, we may elect with various newly developed software packages such as National Instrument's LabView™ and Measure™ to pass sensor output to selected spreadsheet cells that double as the inputs and outputs of a STANN. The self-training net may then simultaneously train on any desired input output relationship as data originates from the system of interest. This development now paves the way for totally adaptive control systems, entirely implemented within Microsoft Excel. In theory, it should be possible for a STANN to observe the action of

human operators in restoring various production systems back to their intended set points. Then, after sufficient 'apprenticeship' period, they may step in to control that process by outputting the necessary DDE control parameter adjustments.

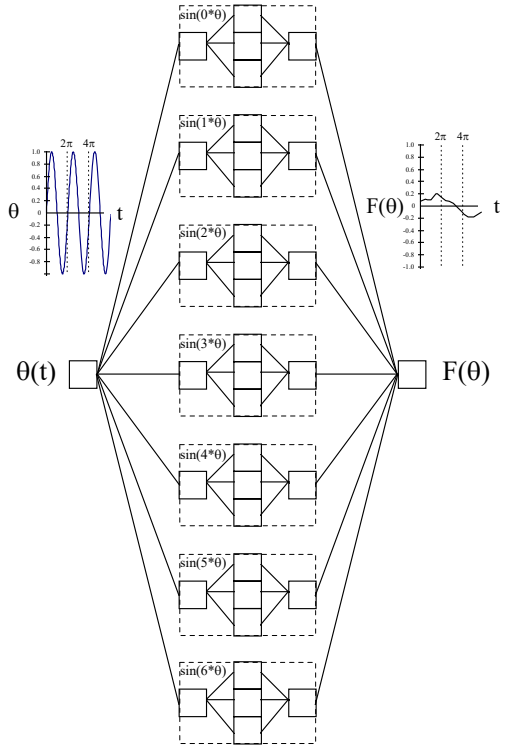
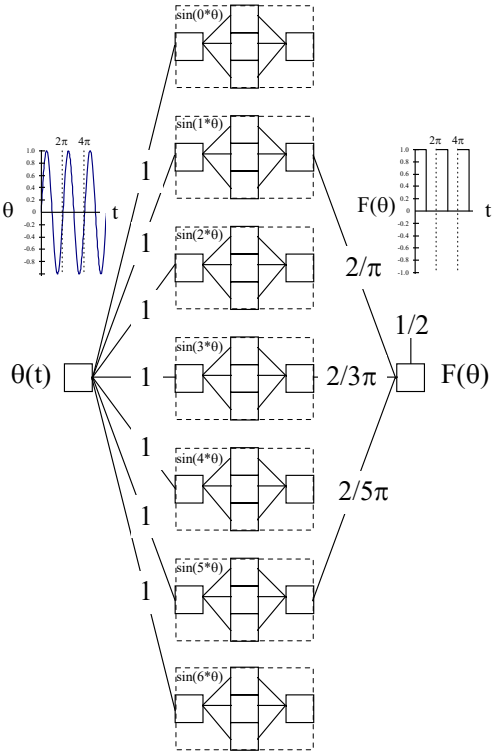


Figure 6. Fourier analysis network prior to training.



$$F(\theta) = \frac{1}{2} + \frac{2}{\pi}\left(\sin\theta + \frac{1}{3}\sin 3\theta + \frac{1}{5}\sin 5\theta + \ldots\right)$$

Figure 7. Fourier analysis network following training. Note the zeroing of odd harmonic components.

4. System Prototyping and Self-Organization - As mentioned above, one of the most powerful features of the STANN is that the transfer function of each processor element is free to take a wide variety of forms. Therefore, it need not be restricted to an analytical and differentiable squashing functions such as the sigmoids popularly used in ANN construction (see for instance, Skapura & Freeman, 1991). Individual processing units may even take the form of whole ANNs representing the input-output response of various analog and digital devices. Training of a collection of such composite ANN systems would be tantamount to the discovery of just how to connect such subsystems to attain some desired global response. Thus the trained connection weights between various electrical devices such as diodes, transistors, etc. would represent connecting resistances to attain some overall electronic function. This device prototyping concept is extendible to other digital electronic applications involving such components as flip-flops and counters. It would likewise be applicable to a wide range of optical, mechanical, and hydraulic systems.

As an example of such device prototyping I have performed preliminary experiments in which I have coupled various ANNs representing various harmonic generation devices in parallel, driven by a single sinusoidal signal generator to produce some predetermined wave form. In the instance shown in Figures 6 and 7, the compound network self-organizes itself to allow conversion of the sinusoidal input into a rectangular pulse output. We note that entirely consistent with the Fourier result, connection weights to the even harmonic devices nullify themselves, while the nonzero weights connecting odd harmonics take on the magnitudes of the Fourier coefficients. In principle, this ANN cascade could represent the prototype of an electrical or optical wave form synthesis device.

5. Potential Applications - In the spirit of object-oriented programming, it is my intention to build Self-Training Artificial Neural Network Objects possessing a variety of innovative properties and methods. Some very broad and powerful applications of such "STANNOs" will include:

- Autonomous Knowledge Base Agents - One of the unique advantages of the STANN is that multiple networks can simultaneously train. Therefore, multiple STANNs may patrol the dynamic data exchange to an Excel spreadsheet, each taking their own unique perspective on the data stream. STANNs or STANN cascades may in turn take action upon that data to extract or manipulate the resulting database.
- Adaptive Creativity Machines - Until now, Creativity Machines have been static in the sense that we have already trained all of its component networks prior to its discovery function. Using the STANN's ability to train on a dynamic influx of data, a Creativity Machine may simultaneously train and create. Presently, I have implemented such schemes allowing multiple STANNs to concomitantly train on the data stream, with a master algorithm periodically copying and pasting the updated networks into their respective positions within a spreadsheet-resident Creativity Machine.
- Adaptive Hardware and Instrumentation - Smart rooms, furniture, vehicles, etc. may utilize the self-trainer to adapt to changing requirements and conditions. Thus an automobile equipped with STANNO control would adapt fuel injection within the driving context to attain the highest fuel economy. Such software could likewise learn driver idiosyncrasies in real time and automatically compensate for disadvantageous motoring habits.
- Artificial Life - Expanding on the metaphor of a "retina," implemented through relative referencing, we may provide various levels of visual processing, analogous to those in human vision, and subsequently, databot reaction to that sensory information. Autonomy derives from the encapsulation of data and function in a manner reminiscent of class objects in object-oriented programming. Further augmenting the databot's organism-like capabilities (e.g., locomotion, reproduction, etc.) the integration of Creativity Machine architectures within the virtual creature's neural cascade allows for its own autonomous movement planning, attention-windowing, and course of action when manipulating its database environment.
- Cooperative Spreadsheet Organisms - Implementing a communication scheme between such spreadsheet objects, it may be possible to achieve cooperative behaviors among databots to build larger neural cascades. Having achieved prototypical structures of this kind in the autonomous construction of Creativity Machines, it is conceivable that much more generalized self-constructing cascades may spontaneoulsy organize within Excel to produce highly sophisticated neural processing structures.

6. Conclusions - The Self-Training Neural Network Object represents a major paradigm shift from the notion of an algorithmic code training an algorithmic neural network simulation. Here, one distributed algorithm trains another, without the intervention of a programmer or sequential computer code. Viewed as a whole, these two reciprocal nets constitute a single, self-training spreadsheet object that need only be physically cut and pasted within the data to learn any patterns contained therein. If data streams through a spreadsheet application by DDE, such STANNOs may similarly glean trends from external agencies and instrumentation.

With the appearance of Excel spreadsheets as a convenient medium of dynamic data exchange from instrumentation, satellite feeds, and the Internet, the STANNO may represent a means for sensing, detection, and prediction, all in real time. Used in conjunction with other cascaded networks and algorithms, such self-trainers will allow for the bi-directional sensing and control required in a number of applied and experimental areas.

7. Bibliography

1. Freeman, J.A. and Skapura, J.A. (1991). *Neural Networks, Applications, and Programming Techniques*, (Reading: Addison-Wesley), 98-99.
2. Kohonen, T. (1982). Self-organized formation of topologically correct feature maps*, Biological Cybernetics* **43**, 59-69.
3. Rowe, J. and Partridge, G. (1993). Creativity: a survey of AI approaches, *Artificial Intelligence Review*, **7**, 43-70.
4. Rumelhart, D.E., Hinton, G.E., & Williams, R.J. 1986). *Parallel distributed processing: Explorations in the microstructure of cognition, Vol. 1* (eds Rumelhart, D.E. & McClelland, J.L.) (Cambridge: MIT).
5. Thaler, S.L. (1994) Musical Themes From Creativity Machine, U.S. Copyright Pau1-920-845.
6. Thaler, S.L. (1995) An Autonomous Discovery Machine for Ultrahard Materials, *Bulletin of the American Physical Society*, Program of the March Meeting, Series II, **40**(1), 592.

7. Yam, P. (1995). *Scientific American* **272**(5), 24-25.