# Neural Network Applications Speed The Navy's Warfighting Ability

*A.J. Maren, R.M. Pap, K.L. Priddy, and R. M. Akita*
*Accurate Automation Corporation*

(1995), Naval Research Review 3, 11-25.

## Editor's Note

*This article describes the successful results of two Navy programs working together. First, funding came from the Office of Naval Research's Cognitive and Neural Science and Technology Division; these funds were then channeled through the Navy's Small Business Innovation Research (SBIR) program to Accurate Automation Corporation (ACC), a small business enterprise in Chattanooga, Tennessee, which has successfully developed novel neural network technologies for the Navy. With the help of the SBIR program, ACC has grown from a basement shop of two people in 1987 to 20 people today in its own building of 10,000 square feet; revenues have grown during the same time from $18,000 to $4,000,000.*

*ACC has developed among other technologies for the Navy "Neural Network Toolbox" and the Sparse MIMD Neural Network Processor, which will make a PC computer work as fast as a super computer.*

*Mentioned in this article are the three phases of funding managed by SBIR to promote small business. In phase 1, up to $100,000 is awarded to a small business for 6 months to evaluate the technical merit and feasibility of an idea; phase II awards up to $750,000 for two years to expand the results of phase 1 by developing a product; and phase III allows for the commercialization of the product through a "buyer" in the government or private industry.*

*Thus by choosing small companies with the right capabilities for the job, the Navy can improve the scientific as well as the business strength of the nation.*

# Introduction

There's an old adage among'st fighter pilots, Speed is life, more is better. These pilots spoke only in reference to their platforms. In today's world, the need for speed applies to all areas of warfighting.

The German army, during the 1940's, mastered the art of "Blitzkrieg," or "lightning warfare," which referred to the technique of rapidly building for and executing an attack. Today, all areas of warfare demand speed and precision. This applies not only to the abilities of the forces in direct contact, but also - and most especially - to functions that support the warrior in combat, including sensor fusion, data communications, database management, target and image recognition, and rapid surveillance. These capabilities directly influence our ability to carry the battle to the enemy, and are the critical elements in our ability to strike an early, crippling blow.

According to Adm. William A. Owens, Vice Chairman of the Joint Chiefs of Staff, "Read the flagship pronouncements of each of the military services: The Army's description of *Force XXI*, the Navy's *Forward ... from the Sea*, the Air Force's *Global Reach, Global Power*, and the Marine Corps' *Operational Maneuver* ... from the Sea. The visions they sketch are remarkably similar. Each points toward the capability to use military force with greater precision, less risk, and more effectiveness. Each relies on three areas of technology:

- Intelligence, surveillance, and reconnaissance,
- Advanced command, control, communications, computers, and intelligence, and
- Precision-guided munitions.

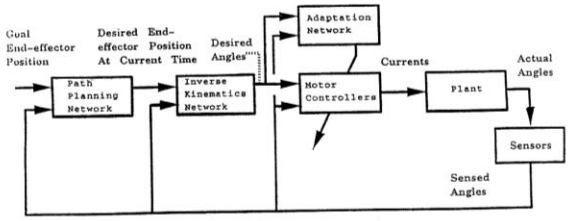
Each recognizes that its efforts are a part of a broader undertaking. I believe that this is the U.S. revolution in joint military affairs."

Neural networks are a key enabling technology that will enhance all three of these critical technologies. Today's battlefield, whether on air, land, or sea, uses a number of neural network hardware and/or software applications. For example, all modems make use of neural network technology for adaptive echo cancellation [1]. Many of the neural network innovations are inspired by biological neural networks. The diversified interplay between different neural network research programs has led to a strong basis for technology development and transition to fielded use by the Navy.

To meet these Navy needs, we have been developing novel neural network technologies that will support and speed the Navy's warfighting capabilities on many levels. These technology developments, sponsored by the Navy Small Business Innovation Research (SBIR) program, include neural adaptive control (leading to advanced flight control methods), sensor fusion and figure-of-merit determination as well as data compression and automatic target recognition. Neural networks apply to Militarily Critical Technologies [2] such as sensor fusion and signal processing, hypersonic / waverider

**Figure 1.**

*Overall controller system design.*




aircraft design and control, simulation / visualization methods, and intelligent processing equipment. In the latter area, in order to develop a platform that can give the Navy maximal computational speed, Accurate Automation Corporation (AAC) has developed a Multiple Instruction, Multiple Data (MIMD) *Neural Network Processor* (NNP™).

The following is a summary of AAC's recent developments in neural network technology for diversified Navy applications.

# Neural Adaptive Control

An autonomous control system is one in which the system itself generates the appropriate control action and/or trajectory. An autonomous controller, whether used in robotics or for flight control, should use desired position values in relation to current position to determine the appropriate motion within a range that is dictated by the capabilities of the plant. Adaptive control systems are those systems that change their own parametric structure to compensate for changes in the plant being controlled.

As part of ongoing work in adaptive control, AAC has developed novel neural network methods for inverse kinematics determination, under a Phase II SBIR contract funded by the Office of Naval Research (ONR). Our *Neural Network Processor* has been used to solve the inverse kinematics problem in near-real-time. These solutions were tested on a real robot using a VME card cage and a dual DSP (TI-TMS 320-C40) implementation attached to the *Neural Network Processor*. The tests were run at NASA Marshall Space Flight Center, AL, using the Proto Flight Manipulator Arm or PFM. In addition to the inverse kinematics using neural networks, a unique joint controller [3] was developed using the functional link neural network paradigm. The overall concept of this is shown in Figure 1.

The inverse kinematics problem can be explained as a set of coupled equations which couple joint parameters to the desired end effector trajectory. Thus, by solving this set of

coupled equations we can determine the desired joint moves to obtain the desired trajectory. The solution of optimization problems has been found using recurrent networks [4-8] for a variety of applications. We will show how the solution of sets of equations using a linear Hopfield network can also be modified to solve the inverse kinematics problem.

Classical manipulator kinematics describes the position of an end effector based upon the positions of the joints of a robot arm. Generally, this is given by a set of non-linear equations, f(.) in joint space, (θ).

$$y(t) = f(\theta(t)) \tag{1}$$

What we really want to know in the inverse kinematics case are the desired joint positions, (θ), to obtain y(t). The inversion of f(.) is difficult due to the dimensionality, multiple joints, and the inherent non-linearities found in robot motors.

$$\theta(t) = f^{-1}(y(t)) \tag{2}$$

A common method to facilitate the solution of the inverse kinematics problem is to linearize the forward kinematics, i.e. to differentiate f(.) with respect to time, yielding the velocity equation (or differential kinematics)

$$\frac{dy}{dt} = \frac{d}{dt}[f(\theta)] = J(\theta)\frac{d\theta}{dt} \tag{3}$$

where J (θ) is the Jacobian of θ.

The prescribed trajectory y(t) is then tracked by a linear approximation via inversion of the linear velocity equation and integrating the obtained joint angle velocities.

Using this approach, the inverse kinematics problem reduces to the inversion of the Jacobian matrix J[θ(t)] at all time instants along y(t), yielding

$$\frac{d\theta}{dt} = J^{-1}(\theta)\frac{dy}{dt} \tag{4}$$

The solution for the inversion of the Jacobian is obtained using a combination of a feedforward neural network and a linear Hopfield network. The Hopfield neural network solves an equation of the form

$$x(i+1) = Wx(i) + u \tag{5}$$

where

W is a symmetric (n x n) real or complex weight matrix,

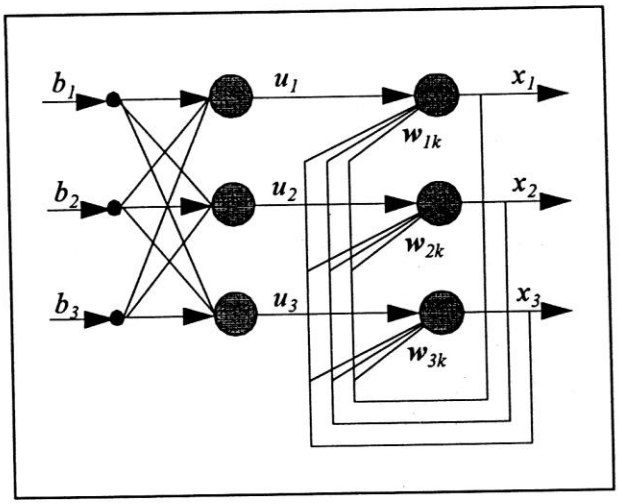
u is a real or complex n-vector of inputs, and

x(i) is the ith iteration of the n-dimensional vector of neuron states.

The resulting equation converges to a solution when the spectral radius, ρ(W), is less than unity, i.e. the eigenvalues of

**Figure 2.**

*A hybrid linear dynamic network, comprised of a feedforward layer followed by a linear Hopfield network.*




W lie within the unit circle in the complex plane. When solving equations of the form Ax = b, the required finesse is to set the weight matrix and input for the Hopfield network to

$$W = I - \alpha A^H A \tag{6}$$

$$u = \alpha A^H b \tag{7}$$

where

$A^H$ is the Hermitian (complex conjugate transpose) of A.

These equations converge for

$$0\,\alpha\,\frac{2}{\rho(A^H A)} \tag{8}$$

In order to speed computations the spectral radius is often replaced by the trace which is the upper bound for the spectral radius.

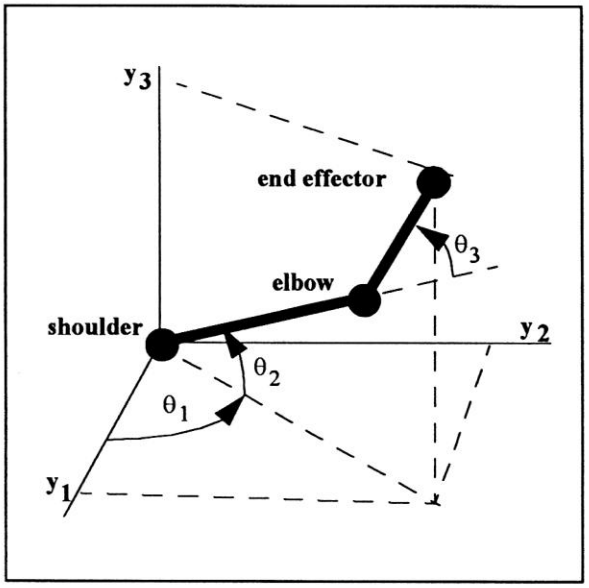
$$0 < \alpha < \frac{2}{trace(A^H A)} \tag{9}$$

The neural network implementation of the linear Hopfield solution for a system of equations of the form Ax = b is shown in Figure 2.

The inverse kinematic problem is solved by forming a difference equation for (t) at a given discrete time along the

**Figure 3.**

*The robot manipulator model used for analysis of the inverse kinematics problem.*




y(t) trajectory. The required solution is then used to move the joints to the desired position until the next time increment. Typically, the solution is obtained in a few hundred iterations of the Hopfield network on an NNP which is much faster than real-time to a robot joint.

$$\Delta\theta\,(i+1) = W\Delta\theta\,(i) + u \tag{10}$$

where

u is a real or complex n-vector of inputs held constant at time t.

$\Delta\,\theta$ (i) is the ith iteration of the n-dimensional vector of neuron states for time t.

The first step is to determine the values for the weight matrix and the input, u, for the Hopfield neural network. From the previous explanations it is fairly straightforward to see that the weight matrix and input to the Hopfield network for the linearized method of computing the desired joint positions are given by

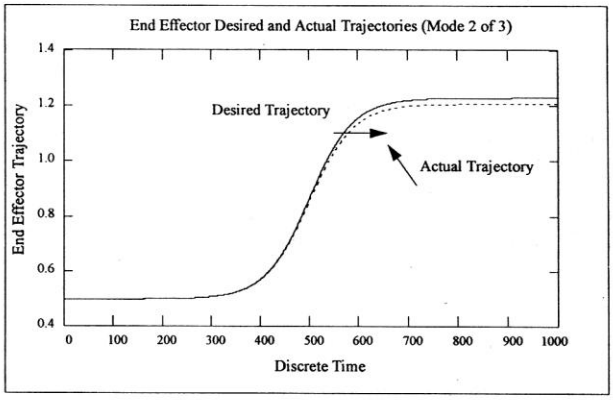
$$W = I - \alpha J^H J \tag{11}$$

$$u = \alpha J^H b \tag{12}$$

$$0 < \alpha \frac{2}{trace(J^H J)} \tag{13}$$

**Figure 4.**

*Desired and actual end-effector trajectory in Cartesian space (1000 points). The error of mode 2 vanished when the trajectory consisted of 1500 points.*

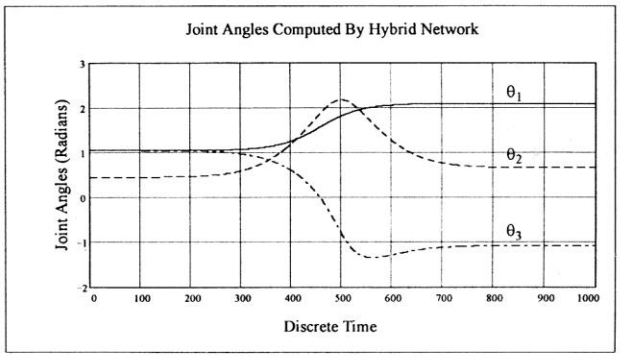



which yields the "best" solution for $\Delta\theta$ for each time step in a least squares sense. Once we find the solution for $\Delta\theta$, we issue incremental changes to the joints which maintains the desired trajectory.

The hybrid feedforward-Hopfield neural network is capable of solving the inverse kinematics problem in real-time when implemented on the AAC *Neural Network Processor* (NNP$^{TM}$). The Hopfield neural network was shown to converge to an optimal solution in a least squares sense and is applicable to a variety of optimization problems.

**Figure 5.**

*Joint angle trajectories computed by the hybrid feedforward/linear Hopfield network.*



This capability simulated the motion of a three joint robot arm on a Silicon Graphics 4D380VGX superminicomputer as depicted in Figure 3. The arm was given a straight line trajectory in Cartesian space consisting of 1000 points and 1500 points. The trajectory for the 1000 point case was identical to the desired with one slight exception in one of the dimensions as depicted in Figure 4. The 1500 point case was exact in all three dimensions. The joint angles generated by the hybrid neural network are shown in Figure 5.

As a result of the success in the inverse kinematics project, the concepts were applied to flight controls for LoFLYTE.

# Neurocontrol for the Low Observable Flight Test Experiment (LoFLYTE) Aircraft

The Low-Observable Flight Test Experiment (LoFLYTE) Advanced Technology Testbed Aircraft is being built to demonstrate neural network technologies in a real world aircraft using some of the Navy-funded SBIR research at AAC. LoFLYTE is laying the foundation for developing next generation aircraft. The LoFLYTE hypersonic aircraft, shown in Figure 6, is a research test vehicle to investigate performance of a waverider aircraft shape at hypersonic velocities (regime of Mach 5). A waverider is an aircraft which rests on its own shock wave as it flies. Waveriders such as LoFLYTE would typically utilize SCRAMJET engines.

Hypersonic aircraft, such as LoFLYTE, could be used in a number of applications. First, they could be configured as an unmanned surveillance platform. Due to their high speed, they will be able to make passes over areas with minimal concerns about being shot down. This will greatly increase our surveillance capabilities during hostilities. Second, a hypersonic cruise missile could be configured with a capability to reach enemy targets much faster than conventional cruise missiles. The need for rapid attacks is supported by A. Fields Richardson (Capt. USN, Ret.), who, during Desert Storm, served as Principal Navy Liaison to the Joint Forces Air Component Commander and head of the Navy Strike Cell. According to Mr. Richardson, "The ability to strike quickly and with great precision is critical to tactical and strategic success. To launch a cruise missile or an aircraft at a target 1,000 miles away and receive a Battle Damage Indication (BDI) report in 20 minutes will provide dramatic tactical advantage, enabling an irresistible build-up of momentum for the force possessing that capability."

Finally, hypersonic vehicles could be configured as a manned aircraft to be launched from the decks of aircraft carriers.

The LoFLYTE project, funded from an Air Force Phase II SBIR Contract, is the focal point and primary demonstration vehicle for several Phase I and Phase II hypersonic-related SBIR contracts funded by the Air Force and by NASA.

The primary focus of AAC's ONR-sponsored research, as it relates to this project, has been to apply lessons learned from biological neural systems to accurate real-time motor control. This research has led to a number of technologies applicable to the real-time control of complex systems, ranging from robotic manipulators to helicopters to hypersonic aircraft.

Real-time control of such complex systems has required not only new, rapidly adaptive neural network algorithms, but also hardware which can carry out the necessary computations with great speed. The ONR-sponsored work has led directly to development of the AAC *Neural Network Processor* (NNP$^{TM}$) discussed in the next section. This processor makes possible the extensive and rapid computations necessary to control a hypersonic aircraft that will fly at Mach 5. In addition to the NNP$^{TM}$, AAC has developed, partially under ONR sponsorship, a full Toolbox of neural networks and learning methods. Several Toolbox neural networks, including the Adaptive Critic [9], have been successfully applied to the control of complex systems at AAC. The Toolbox is central to the development of the control algorithms being used in LoFLYTE. The motor control and motor-mapping algorithms developed for ONR are other examples of Navy technology which have formed a basis for portions of the LoFLYTE aircraft control design.[

AAC has developed neural network control algorithms inspired by drive-reinforcement theories of animal learning and adaptivity. We have applied these methods to the control of electrical motors in robotics tasks. By building upon these algorithms, we have created an adaptive actuator controller for LoFLYTE which adapts to changing loads on LoFLYTE's tiperons and rudders. We have built upon insights into how the

**Figure 6.**

*The AAC LoFLYTE waverider aircraft.*

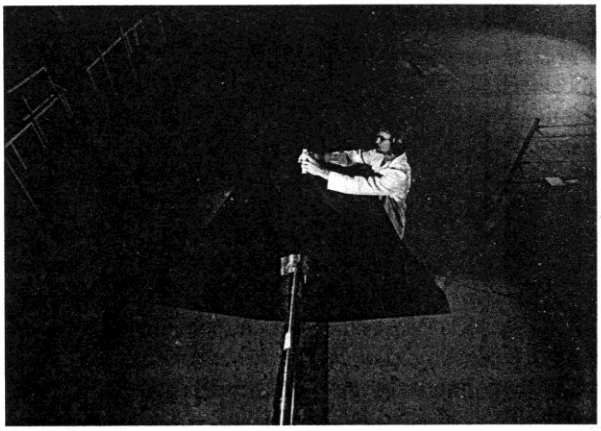


**Figure 7.**

*The AAC Neural Network Processor (NNP$^{TM}$).*

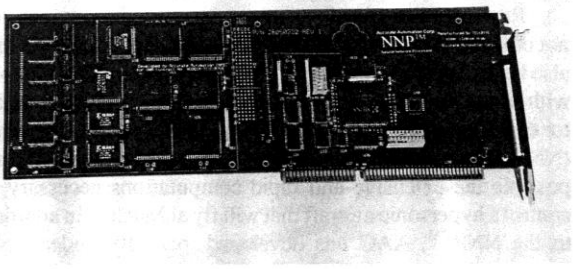
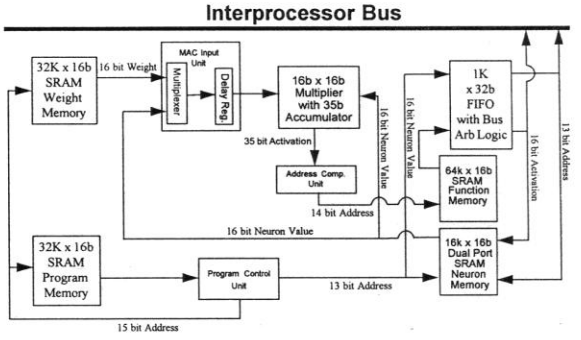


**Figure 8.**

*Block diagram and interprocessor bus architecture for the AAC NNP$^{TM}$.*




human brain maps control objectives to desired joint motions to develop new neural network-based methods for mapping flight control objectives to actuator commands.

# Multiple Input, Multiple Data (MIMD) Neural Network Processor

As an ongoing part of the research conducted under sponsorship by the Naval Research Laboratory and by Naval Command, Control, and Ocean Surveillance Center, RDT&E Division, Accurate Automation has developed a digital Neural Network Processor (NNP$^{TM}$) which is capable of true parallel processing.

The underlying philosophy in the design of the sparse Multiple Instruction Multiple Data (MIMD) NNP$^{TM}$ has been to achieve maximum computational efficiency in both a single processor and multiprocessor environment by optimizing the design to compute neuron values very efficiently [10, 11]. This is in stark contrast to previously proposed neural network processors which are typically based on classical Single Instruction Multiple Data (SIMD) matrix/vector multiplication architectures. Our design fully exploits the intrinsic sparseness of neural network topologies. Moreover, by using a MIMD parallel processing architecture, one can update multiple neurons in parallel with efficiency approaching 100% as the size of the neural network increases.

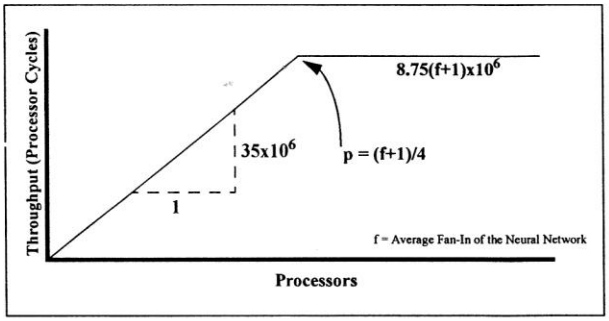
To achieve the desired efficiency we have adopted a design which: 1) Uses an instruction set optimized for neural network processing, allowing one to compute a neuron activation without arranging the weight matrix into linear arrays and/or inserting artificial zero-weighted connection 2) Uses a MIMD parallel processing architecture to permit neurons with totally different input topologies to be updated simultaneously without loss of efficiency; and 3) Uses dual neuron memories to virtually eliminate memory contention and maintain absolute memory coherence.

The NNP$^{TM}$ (see Figure 7) is capable of implementing 8K total neurons with 32K interconnections per processor. A fully configured PC version of the NNP$^{TM}$ is capable of interconnecting 8 modules for a total of 8K neurons and 256K interconnections. In addition to the PC version, the VME version is mated with two TI TMS320C40 DSPs which allow for real-time data manipulation and processing. The VME card is capable of stacking three modules with additional modules requiring a separate VME card due to power limitations. The NNP$^{TM}$ design is based upon a linked list concept which allows any neuron in the network to be connected with any other neuron. Thus, any of the recurrent networks, such as the Hopfield network, or any feedforward networks such as the multilayer perceptron, can be easily implemented using this processor. This hardware capability is vitally important in control applications because it gives us a neural computation engine which is easily adapted to changing control inputs and boundary conditions.

A block diagram of the NNP$^{TM}$ architecture is given in Figure 8. The program instructions and associated weights, as needed, are stored in memory. For a given operation, the instruction is decoded and the necessary value fed from neuron memory to the multiplier input unit along with the weight value the two values are fed to the Multiply ACcumulator (MAC), and the result is passed as an address to the function memory. The address is then used to fetch the appropriate value from the transfer function. This result is then passed though a First-In-First-Out (FIFO) unit and stored in the buffer memory for each of the modules via the interprocessor bus. When an "interchange neuron and buffer memory" (inbm)

**Figure 9.**

*NNP$^{TM}$ throughput as a function of additional processors.*




instruction is encountered, each processor finishes processing its code segment before the buffer and neuron memories are interchanged. Once the memories are interchanged, processing continues until another inbm or stop instruction is encountered. The processor also has the ability to multiply two neuron values together. This is done by passing the previous neuron value encountered to the MAC as an input and then passing the current neuron value as an input to the MAC followed by a multiplication operation. The result is passed through the transfer function and stored in buffer memory as explained previously.

The fully pipelined implementation of the neural network architecture delivers nearly one instruction per cycle which allows the neural network board to execute nearly 35 million instructions per second. Using the standard definition of a connection, a byte-wide multiply-accumulation, the NNP$^{TM}$ yields over 140 million connections/sec for a single board. As additional modules are added, the speed increases linearly with over one billion connections per second possible with a full complement of eight processors.

One of the most vital aspects of parallel computing is the ability of the architecture to maintain memory coherence. The AAC NNP$^{TM}$ accomplishes memory coherence through the use of two separate memories, combined with a special interchange instruction. In most neural network implementations, the results from one layer are multiplied by a series of weights summed, and then passed through a transfer function, typically a sigmoid function, before being used as an output for the neurons on the next layer.

In the NNP$^{TM}$, the inputs are read from neuron memory, with the outputs from the transfer functions stored in buffer memory. When all of the neurons on a layer have been processed an interchange buffer and neuron memory instruction is issued which makes the new data available for use by the next layer of neurons.

A particular NNP$^{TM}$, in a worst case scenario, takes four clock cycles to access the bus and write a neuron value to the buffer memory. On the average, each processor would need to access the bus every (f+1) clock cycles, where f is the average fan-in to a given neuron in the network. Thus, the number of processors allowed before contention occurs is $p < (f+1)/4$. When $p > (f+1)/4$, then bus contention occurs. A depiction of the NNP$^{TM}$ throughput as additional processors is added, shown in Figure 9. As can be seen in the figure, the throughput increases linearly as processors are added until $p > (f+1)/4$.

The inherent speed of the NNP$^{TM}$ in processing sparse matrices makes it ideal for computing neural network structures such as the cooperative-competitive neural network, described in the next section.

# Sensor Fusion

One of the primary objectives of our work has been to develop new sensor data fusion capabilities for the Navy. Sensor fusion is a Militarily Critical Technology [12-14]. Neural networks present a method by which sensor fusion systems can learn from experience, instead of always requiring explicitly the *a priori* probabilities that are currently needed for existing (e.g. Bayesian) formalisms. Further, neural networks have the potential to give a system adaptability to changing environments and conditions. Finally, neural networks can be implemented in exceptionally fast parallel-processing hardware (such as the AAC NNP$^{TM}$), thus overcoming the huge computational burden associated with real-time sensor fusion. Such a neural network-based capability can play a crucial role in enabling the Navy to build integrated systems, linking together systems which are currently "stovepiped."[

One of the big challenges in sensor data fusion is assigning each target to the right track. Gates can be used to find out what targets are in the vicinity of the expected target positions from different tracks. There is still a problem of conflict resolution, when different targets could be assigned to two or more tracks. We have addressed this challenge by developing the novel COoPerative-COMpetitive (COPCOM) neural network. The neural network determines which items in a given Set A have closest similarity to items in another Set B. This network operates by making iterative target-to-track assignments, so that:

- Targets are assigned where there is maximal closeness between a target and the track across a set of matching metrics, and simultaneously
- Targets are assigned where there is minimal conflict between a prospective match and other possibly competing matches.

In this, it offers a more robust approach to assignment than simple non-optimal assignment methods, and unlike the existing optimal assignment algorithms, it can be implemented in

parallel-processing hardware (e.g. the AAC NNP$^{TM}$) for real time solutions to the target-to-track assignment problem [15].

The advantage of using the COPCOM network for assignment tasks is that it makes its first (highest strength) assignments to those matches which have the overall highest values in favor of making the assignment (cooperation across multiple dimensions of similarity), and which also have the least competition with other possible assignments. By making the least ambiguous assignments first, the complexity of the overall problem is reduced. This can make it easier to determine future assignments.

The multilayer COPCOM neural network was initially developed to deal with one of the major challenges of image understanding - that of recognizing objects when they are composed of many different parts, are partially obscured, and/or have specular reflections. To meet this challenge, an early, prototype version of the COPCOM neural network identifies those portions in a segmented image which are most related to each other. This early version of the COPCOM neural network has been applied to images where high specularities, dark contrasting shadows, and including objects present substantial challenges for image understanding [16,17]. The COPCOM neural network has since been redefined to create associations between objects in two different sets. This can be used for matching objects, or tracking the evolution of a multipart system over time.

Inspiration for the COPCOM design came from the perceptual psychology of vision, which suggested that many factors, e.g. similarity of intensity, boundary line continuation, and proximity, all played a role in perceptual organization [18,19].

The COPCOM neural network plays a vital role in target-to-track assignment in the AAC Sensor Fusion Tracking System. Preliminary coarse and fine gating produce a string of potential new-target matches for every Master Target Track (MTT). The COPCOM method is then used to resolve matching assignments. A matrix of possible matches to possible tracks is used to partition out the problem, so that only the subset of new detections which can potentially match a given subset of targets is considered at a time. Unique target-to-track assignments are made before the COPCOM network is used. A search of the COPCOM output nodes for those whose activations pass threshold yields an ordered list of non-competing assignments. The pairwise combinations with the strongest activations are listed first. The Tracker uses this output to make assignments, prune the remaining possible target-to-track assignment possibilities, and rerun COPCOM as often as necessary to get a complete set of assignments.

The cooperative-competitive method has been demonstrated to be effective for target-to-track association, even in dense target environments. Some of the scenarios for which COPCOM effectiveness has been shown include crossing targets, splitting targets, and dense targets (e.g. close groups of 4, within overall interacting scenarios of 16 proximal targets) [20-22].

The items to be matched, i.e., the members of Set A and Set B (e.g., new detections and tracks), must have the same dimension or vector length, denoted N, and the elements in these two vectors should have the same meaning. (Of course, if matching is being done between elements of the same set, then this requirement is automatically satisfied; Set **A** = Set **B**.) For example, the items of Set **A** and Set **B** could each be the x and y positions of objects in Euclidean 2-space. We denote the nth dimension, of the ith item of Set **A** as $a_n^i$, and the nth dimension, of the jth item of Set **B** as $b_n^j$. Let there be a total of I items, i = 1..I in Set **A**, and J items, j = 1..J, in Set **B**. The COPCOM network operates on functions of the distance between the vector components for each possible pairwise match of members of Sets **A** and **B**, over each of the N dimensions describing each set member.

The COPCOM network conceptually consists of four or more layers. A basic COPCOM network is shown in Figure 10. The nodes in Layer 1 represent the individual items themselves. The nodes in Layers 2 and above represent the strength of relationships between pairs of items taken from Set **A** and Set **B**, not to the individual items themselves. This means that if Set **A** has I items, and Set **B** has J, there are I*J nodes in each of N subnets at the second and succeeding layers, to accommodate that number of pairwise relationships.

COPCOM works by assessing relative similarities between items across multiple dimensions. In Layers 2, 3, and subsequent intermediate layers, there is a separate subnet for each dimension which will contribute to the overall assignment decision. Thus, if the items in Sets **A** and **B** can each be described by a 2-D vector (e.g., x and y values), then Layers 2, 3,... will each have two subnets; one for each dimension. We could call these the X subnet and the Y subnet. This paradigm

**Figure 10.**

*The CoOPerative-COMpetitive (COPCOM) neural network architecture.*