(4)

NSWC TR 90-169

NEURAL NETWORK IMPLEMENTATION OF AN F-14 BATTLE MANAGEMENT FUSION ALGORITHM RULE BASE

BY GEORGE ROGERS JEFFREY L. SOLKA DAVID STEFFEN
STRATEGIC SYSTEMS DEPARTMENT

JUNE 1990

NAVAL SURFACE WARFARE CENTER
Dahlgren, Virginia 22448-5000 • Silver Spring, Maryland 20903-5000

NSWC TR 90-169

NEURAL NETWORK IMPLEMENTATION OF AN F-14 BATTLE MANAGEMENT FUSION ALGORITHM RULE BASE

BY
GEORGE ROGERS
JEFFREY L. SOLKA
DAVID STEFFEN
STRATEGIC SYSTEMS DEPARTMENT

JUNE 1990

NAVAL SURFACE WARFARE CENTER
Dahlgren, Virginia 22448-5000 • Silver Spring, Maryland 20903-5000

FOREWORD

The Attack and Defense of Maritime Resources in Adverse Locales Simulator (ADMRALS) system is a distributed multiwarfare simulation system. Central to the outer air battle in ADMRALS is the F-14 battle manager, currently consisting of a rule base of 722 rules. This has been implemented using a multilayer perceptron network. A hybrid σ-π network has been devised for implementation on a parallel computer.

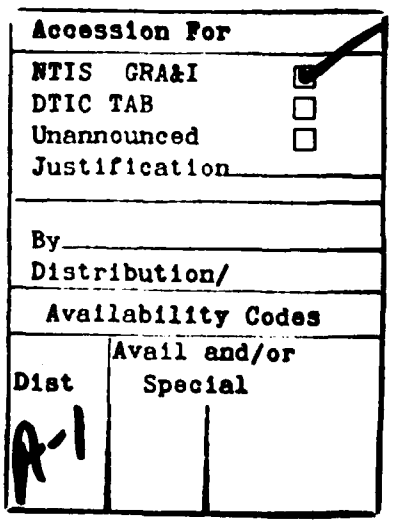
This study was funded by the Warfare Systems Architecture and Engineering Effort and was conducted within the Space and Ocean Geodesy Branch of the Space and Surface Systems Division.

This technical report was reviewed by Patrick E. Beveridge, Head of the Space and Ocean Geodesy Branch and J. Ralph Fallin, Head of the Space and Surface Systems Division.

Approved by:

J. Ralph Fallin

R. L. SCHMIDT, Head
Strategic Systems Department



CONTENTS

ILLUSTRATIONS

TABLES

INTRODUCTION

The Attack and Defense of Maritime Resources in Adverse Locales Simulator (ADMRALS) system is a distributed multiwarfare simulation system that allows variable fidelity/complexity modeling of engagements between user designed naval battle force architectures (existing or proposed) and large-scale threat scenarios. ADMRALS utilizes TCP/IP, a heterogeneous processor suite, and a locally developed distributed network environment to give concurrent processor capability to a systems level multiwarfare naval simulation. It is believed that ADMRALS will fill the apparent "systems level" void existent in multiwarfare analysis capabilities within the Navy and provide a sound foundation for potential evolutionary growth into a key system test and analysis facility for the Navy. The first warfare area investigated was that of naval air defense. Central to the naval air defense role is the F-14, and central to a simulation of this type is a battle management algorithm for the F-14s.

The F-14 Fusion Algorithm Rule Base section presents the rule base that makes up the F-14 battle management data fusion algorithm as implemented for a May, 1988 demonstration. Ideally, this type of algorithm should be fault tolerant. That is, if a set of input conditions should be outside of the rule base, the algorithm should return a reasonable result rather than forcing termination of the simulation. Neural networks are inherently fault tolerant and as such are logical candidates for algorithms of this type. The type of network deemed most suitable for this application is the multilayer perceptron.[2] The applicable feed forward and back propagation equations are presented in Multilayer Perceptions with Error Back-Propagation.

The network Design and Implementation section discusses the network design and implementation of a set of subnetworks, a single large network, and a single σ-π hybrid network that eliminates the need for any conditional (if-

The Network Performance section begins with an evaluation of the performance for both the subnets and the relative execution times to be expected on a parallel computer. The conclusions and some closing remarks are presented in the last section.

F-14 FUSION ALGORITHM RULE BASE

The F-14 battle management fusion algorithm has the function of combining the current values of the Current Activity (CA), the Prioritized Target List (PTL), the Previous Event List (PEL), and the Communications (COMM) List, and returning a current request from the Current Request List (CRL). The possible values for each of these groups are given in Table 1.

The CRL value is passed to a scheduler routine which determines whether that CRL is possible. For example, if the CRL is "Attack," but the F-14 is out of weapons, the scheduler will override the "Attack" value. The output of the scheduler is a new current activity for the next time step. The scheduler is not included in the neural network.

The rule base consists of 26 rules, which are listed in abbreviated form in Table 2. Table 3 gives an example expansion of one of the rules. Note that each rule can contain more than one CA, PTL, etc. The result is that these 26 rules are equivalent to 722 input cases or training patterns.

This rule set was intended only to cover a specific aspect of the outer air battle, namely that of a cruise missile attack on a carrier battle group. The cruise missiles can be both submarine and air launched. Although this type of engagement is of fairly restricted nature, it is of sufficient complexity to provide a good demonstration of a neural network implementation in the data fusion role.

MULTILAYER PERCEPTRONS WITH ERROR BACK-PROPAGATION

FEED FORWARD PROCESS

The subnetworks used for the study were multilayered perceptrons with two hidden layers. The output of the j-th node was computed as follows.

TABLE 1. ACTIVITY TABLE

Current Activity

1. Combat Air Patrol
2. Vector
3. Loiter
4. Investigate
5. Surveillance Alert
6. Intelligence Alert
7. Reconnaissance
8. Chase
9. Engage
10. Evaluate
11. Deck Launched Interceptor
12. Stacked
13. Headed Home

Prioritized Target List

0. Null
1. Jamming
2. Engaged Targets
3. Engaged Targets with Jamming
4. Unengaged Targets
5. Unengaged Targets with Jamming

Previous Event List

0. Null
1. Reconnaissance
2. Stacked
3. Loiter
4. Engage and Kill Assessment

Communications List

0. Null
1. Possible Threat or Jamming
2. Bogies
3. Go Home

Current Request List

1. Combat Air Patrol
2. Vector
3. Loiter
4. Investigate
5. Surveillance Alert
7. Reconnaissance
8. Attack
9. Engage
10. Evaluate
11. Deck Launched Interceptor
12. Stacked
13. Headed Home
14. Investigate and Directed Activity (Reconnaissance)
15. Intelligence Alert and Directed Activity (Reconnaissance)
16. Surveillance Alert and Directed Activity (Reconnaissance)
17. Vector and Directed Activity (Stacked)
18. Reconnaissance and Remove Engage Kill Assess

TABLE 2. F-14 BATTLE MANAGEMENT/DATA FUSION RULES

Current Activity	Prioritized Target List	Previous Event List	Communication	Current Request List
1	0	0	0	1
2	0	0	0	2
3	0	0	0	3
11	0	0	0	11
7	0	0	0	7
6	0	0	(0,1)	15
5	0	0	(0,1,2)	5
(1,2,3,7,11)	(1,3)	(0,1,2,3)	(0,1,3)	14
(1-8,10,11)	(4,5)	(0,1,2,3)	(0,1,2,3)	8
(1,2,3,7,8,11)	0	(0,2,3)	1	15
(1-4,6,7,8,11)	(0,1)	(0,1,2)	2	16
(4,5,6)	(0,1,2,3)	1	(0,1,3)	7
4	1	0	(0,1)	4
(1,3,7,8,11)	0	(0,1,3)	3	17
13	(0-5)	0	(0-3)	13
12	(0-5)	0	(0-3)	12
13	(0-5)	2	0	12
(4,5,6,8,10)	(2,3)	0	(0,3)	10
10	1	0	(0,1,3)	14
10	(0,1)	0	2	16
(4,8,10)	0	0	0	7
9	(2,3,4,5)	0	(0-3)	9
9	(2,3,4,5)	4	(0-3)	10
9	(0,1)	(0,4)	(0-3)	10
(1,2)	0	2	0	12
11	0	3	0	3

NOTE: See Table 1 for numerical translations.

TABLE 3. EXAMPLE RULE EXPANSION

EXPANSION OF ENTRY NUMBER 8 OF TABLE 2

Current Activity Choices = (C.A.P., Vector, Loiter, Reconnaissance, D.L.I.)

Prioritized Target List Choices = (Jamming, Engaged Targets with Jamming)

Previous Event List Choices = (Null, Reconnaissance, Stacked, Loiter, Engage and Kill Assessment)

Communication List Choices = (Null, Possible Threat or Jamming, Go Home)

Current Request List Desired = Investigate and Directed Activity (Reconnaissance)

Some example combinations are as follows:

(C.A.P.; Jamming; Null, Null)

(C.A.P.; Jamming; Null, Go Home)

(Vector; Engaged Targets with Jamming; Loiter; Possible Threat or Jamming)

First a weighted sum of its input was computed

$$T_j = \sum_i w_{ij} x_i + \theta_j \tag{1}$$

The output was then computed by applying the logistic function S to this quantity

$$x_j = S(T_j) = \frac{1}{1 + e^{-\alpha T_j}} \tag{2}$$

ERROR BACK PROPAGATION

The input nodes of the network were clamped to a given input pattern and then error terms and edge weight corrections were computed using the standard error back propagation scheme.[2]

First the error for each output node j was computed

$$\delta_j = (x^*_j - x_j) x_j (1 - x_j) \tag{3}$$

Next the error term for each hidden unit j was computed

$$\delta_j = x_j (1 - x_j) \sum_i \delta_i w_{ji} \tag{4}$$

The δ_is in Equation 4 were the error contributions from the nodes below node j in the network. Once all the error terms were computed the edge weights were adjusted

$$w_{ij}(t + 1) = w_{ij}(t) + \varepsilon \delta_j x_i + \eta (w_{ij}(t) - w_{ij}(t - 1)) \tag{5}$$

Weights were adjusted after each presentation of each pattern.

NETWORK DESIGN AND IMPLEMENTATION

GENERAL CONSIDERATIONS

The network design was driven by several factors. First and foremost was the requirement that the network produce the correct answer for each case contained in the rule set. Second was the desire to obtain a network capable of generalizing to unforseen combinations of the inputs. Third and last was the requirement that the network could be easily trained in several hours on the Sun 3/280 minicomputer.

The total number of training cases for the rule set given in the F-14 Fusion Algorithm Rule Base section is 722. The number of cases for each CA varied from a minimum of 24 (for "Stacked") to 70 (for "CAP" and "DLI"). Since each Current Activity (CA) forms its own independent subset of the total training set, and because of the relative training times involved, it was decided to develop a separate subnetwork for each CA.

To get an idea of the relative training times involved, it is instructive to perform the following exercise. The general rule of thumb for training times is that the time scales as the number of patterns cubed. As the number of patterns is increased (one factor), the number of training passes through all the patterns increases (the second factor), and the size of the network increases (the third factor). Let us assume that this rule is strictly valid, and compare the relative training times for 13 subsets of 70 patterns each, with the training time for a single network with 722 patterns. For each of the 13 subsets of 70 patterns, the training time is

$$\tau_1 = (70)^3 = 343000,$$

where the exact time units are machine dependent. The total time to train all 13 subsets is

$$\tau_{13} = 13\ \tau_1 = 4459000.$$

The training time for the total network with 722 patterns is

$$\tau_{722} = (722)^3 = 376367000,$$

which gives a ratio of training times of

$$\rho = \tau_{722}/\tau_{13} = 84$$

This result means that, for the assumptions stated above, the training period for a single large net would take 84 times longer than to individually train each of the 13 subnets.

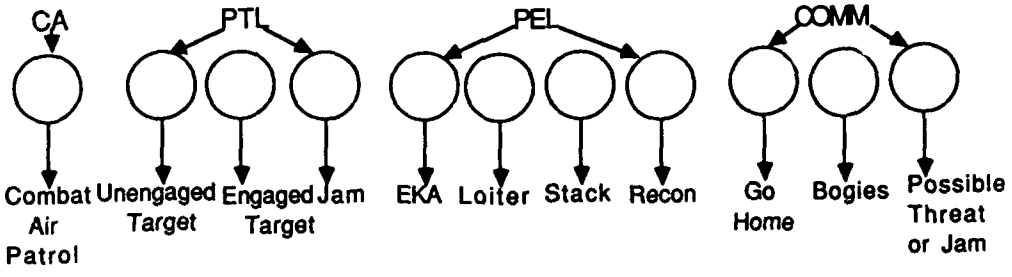
The analysis of the previous paragraph is at best only approximate. However, it is useful for an order of magnitude estimate. Thus, training the total network could be expected to take anywhere from 10 to 100 times longer than training all 13 independent subnets. Each subnet took a maximum of six minutes of CPU time to train. This would scale to between 13 hours and 130 hours of CPU time for a single large network according to the above analysis. In fact, the training time for a single network exceeded 33 hours of CPU time. This clearly shows that using the subnets is desirable from the training aspect.

The sub-networks are described in the following subsection.

CURRENT ACTIVITY SUBNETS

The input scheme for each subnet is given in Figure 1. It consists of one input for the CA, three inputs for the PTL, four inputs for the PEL, and three inputs for the COMM. There is one output for each of the eighteen CRL possibilities. For each current activity stand-alone subnet, the CA input always has a value of one. The arrangement of the neural network for the CAP case is shown in Figure 2. In this figure, the top row of circles represent the input nodes of the network. The second row corresponds to the first hidden layer. Note that only the connections between the inputs and the first node are shown in the figure. Similar connections exist between each of the input nodes and each of the other nodes in the first hidden layer. The third row represents the second hidden layer in the network, where only the connections between the h-11 node and the second hidden layer are shown explicitly in the figure. The bottom row represents the output nodes. Each output node corresponds to one item in the Current Request List of Table 1.

For the other CA cases, the network is of the same size and geometry, the first node just corresponds to the particular current activity for that



N. B. - The presence of a given attribute is indicated by a 1 in the appropriate position.

For example the pattern 11010010001 indicates

Current Activity	= Combat Air Patrol
Prioritized Target List	= Unengaged Target with Jamming
Previous Event List	= Stacked
Communication	= Possible Threat

FIGURE 1. INPUT VALUES FOR THE COMBAT AIR PATROL SUBNET

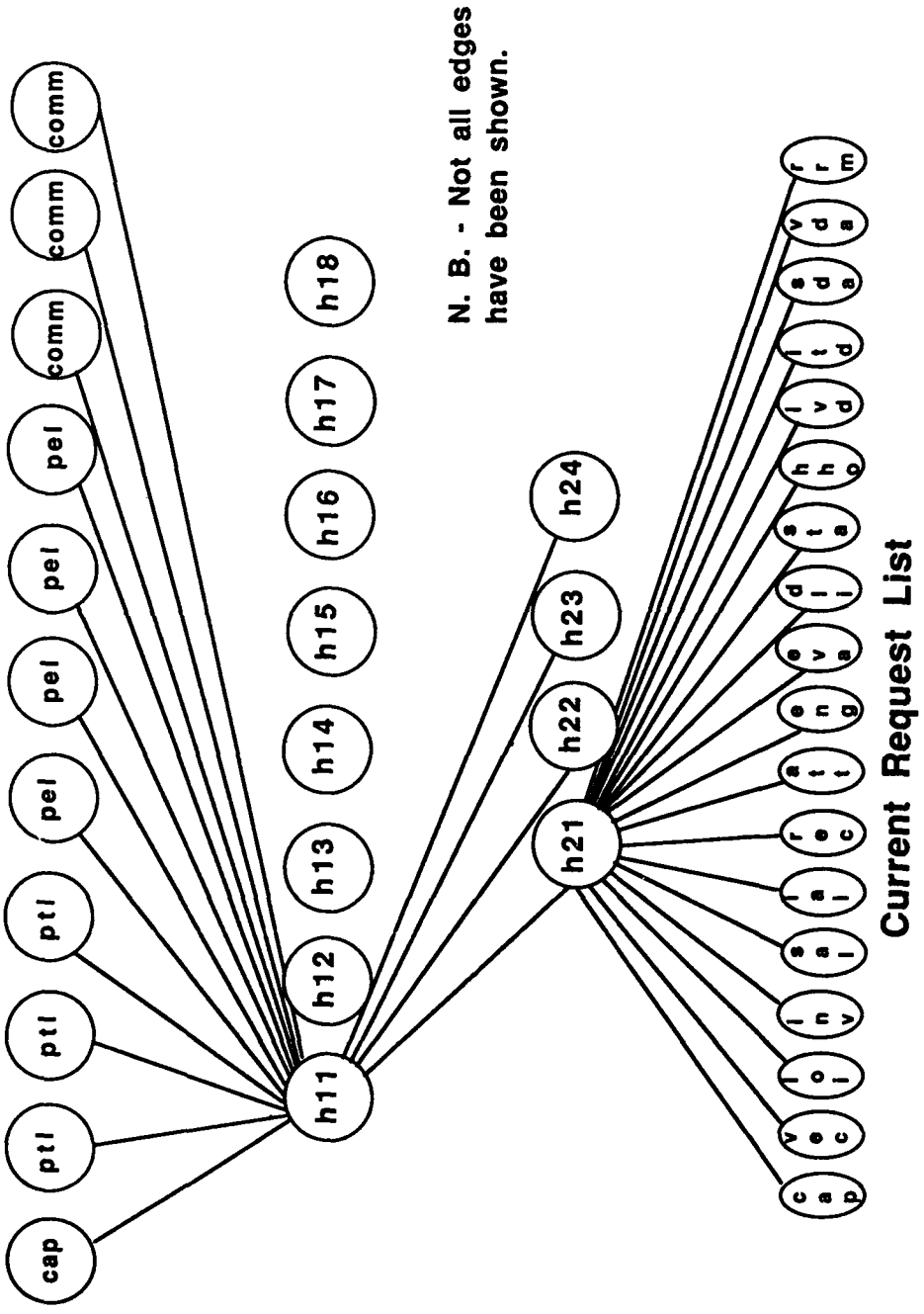


FIGURE 2. SUBNET STRUCTURE

subnet. After training, of course, all the edge weights and offsets will in general be different for each of the subnets.

Although there are 11 input nodes, these do not all represent independent bits of information. The information contained in these inputs can be stored in an eight-bit representation. The network does this in its first hidden layer. It develops its own internal eight-bit representation during the training period.

Similarly, the 18 possibilities contained in the 18 output nodes are expressible in terms of a five-bit internal representation. However, in all of the current activities, less than 16 CRLs are possible for a given current activity. Thus, a four-bit internal representation is sufficient. This is the reason for only four nodes in the second hidden layer.

As can be seen from the analysis of the preceding two paragraphs, the subnet uses a minimum or near minimum number of nodes at each hidden layer for its required internal representation. This has the effect of forcing the network to "discover" and embed the logic of the rule set into its edge weights and offsets.[1] If instead, many more hidden nodes were used, the network could instead merely memorize the individual training patterns, rather than learning the underlying logic. Obviously, generalization to unexpected combinations of the inputs is much more likely to yield a good answer if the logic is embedded rather than if the individual cases are memorized. It should be noted that at present there is no general way to recover the logic directly from the values of the edge weights.

PERCEPTRON NETWORK FOR THE COMPLETE CURRENT ACTIVITY SET

A network consisting of 23 input nodes corresponding to the 13 CAs, 3 PTLs, 4 PELs, and 3 COMMs, was trained on the combined rule set of 722 rules or patterns. The set of output nodes was identical with the subnets. To ensure an adequate number of hidden nodes for the problem, the number of hidden nodes of each subnet were combined to give 104 nodes in the first hidden layer and 52 nodes in the second hidden layer. As expected from the analysis of the General Considerations section, learning was very slow. At the time of writing, the maximum error was only down to 0.30 after 1000 passes and over 33 hours of CPU time. This is to be compared to the training statistics for the subnets presented in Table 4.

TABLE 4. PERFORMANCE OF SUBNETS ON TRAINING PATTERNS

Activity	Number of Patterns	Number of Passes	Rms Error	Max Error
Cap	70	1500	.0044	.0499
Vector	67	1500	.0045	.0606
Loiter	69	1500	.0033	.0373
Investigate	61	1500	.0031	.0259
Surveillance Alert	55	1500	.0029	.0228
Intelligence Alert	60	1500	.0030	.0234
Recon	69	1500	.0041	.0591
Chase	53	1500	.0060	.0911
Engage	48	1500	.0020	.0050
Evaluate	46	1500	.0037	.0364
D.L.I.	70	1500	.0044	.0514
Stacked	24	1500	.0013	.0014
Headed Home	30	1500	.0019	.0062

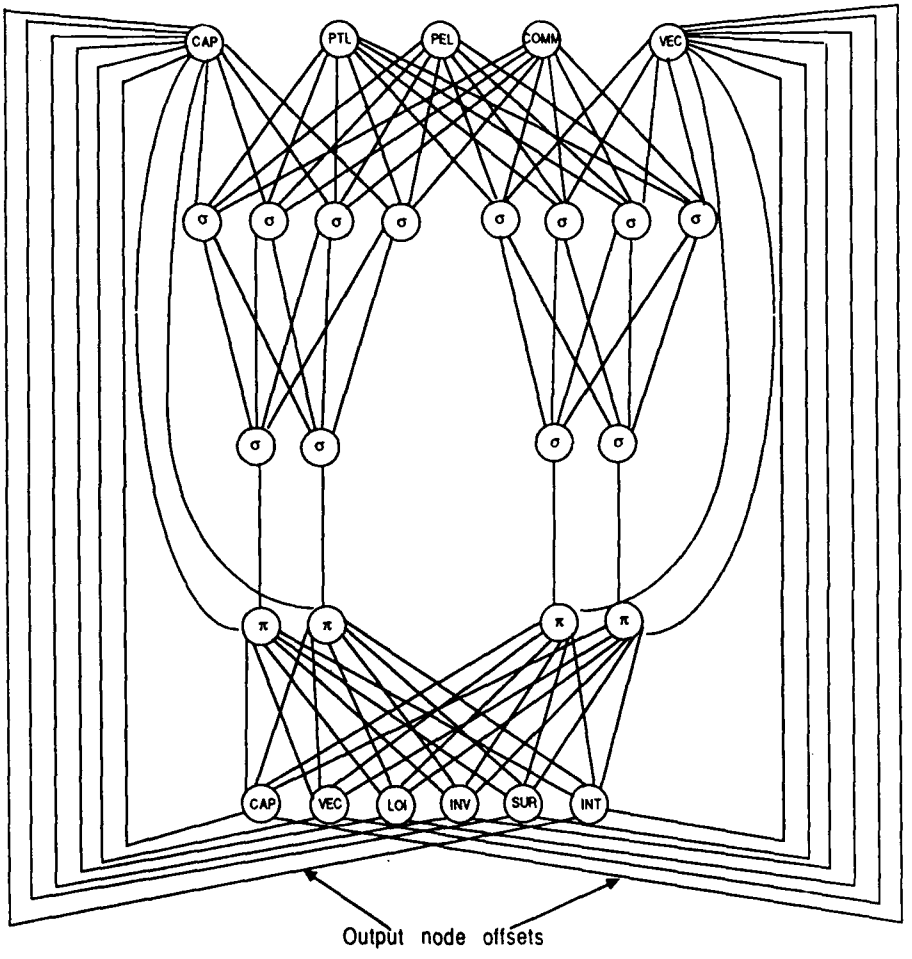
σ-π HYBRID PERCEPTRON FOR THE COMPLETE CURRENT ACTIVITY SET

The 13 subnets can be combined in a σ-π network that completely eliminates the use of conditional tests (if-then statements). On a computer with a high enough degree of parallelism, this would make the computation faster than the original conditional implementation on a serial machine. All of the advantages of the neural network such as generalization would still apply.

The σ-π network discussed below allows the subnets to be trained up separately to the desired accuracy and then combined into a single feed forward network. A simplified σ-π network of this type is shown in Figure 3. In the network depicted in the figure, there are only two current activities ("CAP" and "VEC"), and a single PTL, a single PEL, and a single COMM. The subnets for "CAP" and "VEC" each consist of four hidden nodes in the first layer and two in the second hidden layer. Cross connections between the first and second hidden layers occur only within each subnet, not between subnets. The "CAP" input node is connected to the first hidden layer of its subnet while the "VEC" input node is connected to its subnet. The PTL, PEL, PEL, and COMM inputs are shared between all the subnets with the weights determined by the respective subnet weights. All of the weights and offsets connecting the input nodes to the first hidden layer and the first hidden layer to the second hidden layer are exactly those of the respective subnets. The outputs of the second hidden layer feed directly to the π nodes of the third hidden layer with no cross links. The direct links from the "CAP" node to its π nodes allows those π nodes to act as gates in the following manner.

When "CAP" is on, it sends a one to each π node in its subnet. This turns the π nodes "on" and the output of the "CAP" subnet hidden layers is passed on to the output nodes using the hidden layer to output weights of the "CAP" subnet. Meanwhile, all other current activities ("VEC" in this example) are "off" and pass a zero to their respective π nodes which turns them off. The end result of the π layer is that only the desired subnet makes a contribution to the output layer.

The last remaining detail in the functioning of the net concerns the output node offsets of each subnet. Since they are in general different for each subnet, and the output nodes are shared in this σ-π network, the following scheme is necessary. All of the output node offsets are taken as



N. B. - Not all nodes have been shown.

FIGURE 3. SIMPLIFIED $\sigma - \pi$ NETWORK

zero. The output node offsets of each subnet are used as weights for direct input to output node links. Thus, when "CAP" is on, the "CAP" subnet offsets are fed to the output nodes while the "VEC" subnet offsets are suppressed.

In summary, Figure 3 shows a simplified σ-π network with only two CAs and two subnets. The σ-π network for the F-14 battle manager consists of 13 current activity input nodes, each connected to its own subnet consisting of eight σ nodes in the first hidden layer, four σ nodes in the second hidden layer, four π nodes in the third hidden layer, and eighteen output nodes that are shared with all of the other subnets. The three PTL, four PEL, and three COMM input nodes are also shared by all of the subnets.

As can be seen form the preceding discussion, use of the σ-π network will give exactly the same answers as one would get from the individual subnets. Hence, on a serial machine, such as the Sun 3/280 minicomputer, the subnets can be used in conjunction with logical if statements to ensure that the correct subnet is used for each CA. This can be viewed as a serial implementation of the complete σ-π network. On a parallel machine, it is of course preferable to eliminate all of the serial if tests and use the full network.

NETWORK PERFORMANCE

Each of the CA subnets was trained for 1500 passes through their respective patterns. A learning rate of 0.3 and a momentum transference rate η of 0.8 were used in each case. The current activities with the most patterns ("CAP" and "DLI") each took six minutes of CPU time to train. The other current activities took correspondingly less time based on the number of training cases. The largest error after training was for "Chase" with an error of 0.0911. This was within the training tolerance of 0.1 and so was quite acceptable. Any output value greater than 0.9 is taken as 1.0 and any output value less than 0.1 is taken as 0. This is necessary because of the nature of the logistic function in Equation 2. The training statistics for each CA are presented in Table 4.

Training the "CAP" subnet on all 70 of the "CAP" cases insured that the "CAP" subnet would always give the correct answer (as long as the inputs correspond to one of the 26 rules of Table 2. Similar considerations apply to

the other subnets. Thus, the neural network implementation of the rule base will always give the correct answer, unless the input combination falls outside of the rule base. For the "CAP" case, there are 50 possible combinations of the inputs which fall outside of the rule base. These combinations were input to the "CAP" subnet and the resulting CRL values analyzed. In all 50 cases, the results were reasonable given the input combinations. Most of these combinations do not make sense in terms of the actual implementation. In one case however, the network supplied a good answer to a case that can happen, but had been overlooked in the rule base! This consisted of CA="CAP," PTL=Null, PEL="Stacked," and COMM="Go Home." The network responded with a CRL of "Vector and DA (Stacked)" which is a correct response (CRL="Stacked" would also have been appropriate). This is a graphic example of the advantage of neural networks used in the data fusion mode. The occurrence of an unforeseen input combination does not stop the simulation, but rather gives a reasonable result which allows the simulation to continue.

The relative execution times of both a subnet and the full σ-π network on a highly parallel computer deserve some comment. First, in the full network case, there is an extra π layer of nodes, so it will definitely take longer to execute. On the other hand, the connections going into each hidden sigma node are exactly the same as in the subnet! Hence, if there is a dedicated processor for each node of a given hidden layer, there will be no increase in computation time due to the hidden sigma layers. Each π node has two inputs which are merely multiplied and sent out to each of the output nodes. This is a very fast operation and hence will have only a minuscule effect on the execution time for the network. The only major increase in execution time will then be due to the extra computations occurring at the output nodes! For K current activities, M π nodes, M hidden nodes in the second hidden layer in each subnet, and N subnets, we have M connections coming into each output node for a stand alone subnet and

$$L = MN + K$$

connections coming into each output node for the σ-π network. Thus, the last part of the computation should take approximately

$$\rho = (MN + K)/M$$

or

$$\rho = N + K/M$$

times longer. For the F-14 battle manager, N = 13, K = 13, M = 4, and

$$\rho = [13(4) + 13]/4 = 16.25.$$

If the other layers of the F-14 network are taken into account, there are 11 inputs to each node in the first hidden layer, eight inputs to each node of the second hidden layer, and two inputs to each node in the π hidden layer. There are a total of 65 inputs to each of the output nodes in the σ-π network but only four for each output node in a subnet. The computation at each node is roughly proportional to the number of inputs to the node. (This neglects the time to compute or look up the sigmoid function.) Hence, if there is a separate serial type CPU for each node in a given layer, and if the communication times are neglected, the approximate ratio for a complete feed-forward evaluation is obtained by summing the connections at each layer for each type of network,

$$\frac{\sigma\text{-}\pi}{\text{Subnet}} = \frac{11 + 8 + 2 + 65}{11 + 8 + 4} = \frac{86}{23} = 3.74$$

A complete σ type network was also studied. It consisted of 23 input nodes, 104 hidden nodes in the first layer and 52 in the second, and 18 output nodes. The same analysis has been carried out for this network with the results summarized along with the preceding results in Table 5 and Table 6. The other computation time ratios that result are

$$\frac{\sigma\text{Net}}{\text{Subnet}} = \frac{179}{23} = 7.78$$

and

$$\frac{\sigma\text{-}\pi}{\sigma\text{NET}} = \frac{86}{179} = 0.48$$

Thus, the σ-π network would perform about twice as fast as the σ network in a highly parallel environment and only about four times slower than an individual subnetwork.

TABLE 5. NETWORK CONFIGURATIONS

Network	Inputs	Hidden 1	Hidden 2	π	Output
Subnet	11	8	4	NA	18
Net	23	104	52	NA	18
$\sigma-\pi$	23	(8)(13)	(4)(13)	(4)(13)	18

N.B. - (8)(13) means 8 nodes in each of the 13 distinct subnets within the network.

TABLE 6. NUMBER OF CONNECTIONS FEEDING INFORMATION INTO EACH NODE

Network	Hidden 1	Hidden 2	π	Output	Total
Subnet	11	8	NA	4	23
Net	23	104	NA	52	179
σ-π	11	8	2	65	86

CONCLUSIONS

A multilayer perceptron (MLP) implementation of an F-14 battle management/fusion algorithm has been successfully incorporated into the ADMRALS warfare systems analysis and engineering testbed. The MLP implementation offers several advantages over the use of conditional tests. First, its inherent ability to generalize to unforseen situations compensates for oversights by the rule base engineer. Secondly, revisions of the rule base do not alter the battle management computer code, but only the network's edge weights and offsets. Finally, in keeping with previous studies done with MLPs we would expect the performance of the network would degrade gracefully with perturbations of the edge weights and offsets.[3] The utility of multilayered perceptrons as battle managers has been demonstrated. If a problem can be decomposed into a set of mutually exclusive subcases, as was done here with the current activities, the training of the network is simplified and the training time can be drastically reduced. Whereas on a serial computer the subcases are linked with traditional if tests, on a parallel computer the σ-π network obviates the need for any sequential conditional tests while speeding execution time. While π nodes are well documented,[2] to the authors' knowledge their use as subnet connectors has not previously appeared in the literature. This represents an important innovation in the parallel implementation of MLP neural networks. Using subnets is more efficient in term of execution, reduces training time, and can also be expected to enhance the generalization ability of the network. This is true because each subnet can be "tailored" to its specific subcase, whereas a single large net may offer many more edge weights than necessary and so degrade generalization.

The successful implementation of the MLP F-14 battle management algorithm has produced optimism for future MLP applications. The use of a MLP as an antisubmarine warfare battle manager is being investigated. The direct translation of human knowledge to MLP training patterns is being considered as an alternative to a traditional rule base for complex decision tasks.

GLOSSARY

S = Sigmoid function

T_j = Intermediate result for the j-th node

w_{ij} = Weight from i-th node in layer n to j-th node in layer n+1

x_j = Output of the j-th node

x^*_j = Desired output of the j-th mode

α = Sigmoid slope parameter

ϵ = Learning rate

η = Rate of momentum transference

θ_j = Offset of the j-th node

REFERENCES

1. G. Hinton 1987. "Connectionist Learning Procedures." Carnegie-Mellon University Technical Report CMU-CS-87-115.

2. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. 1986. "Learning Internal Representations by Error Propagation." *In Parallel Distributed Processing*, Vol. 1, by D. E. Rumelhart and J. L. McClelland (eds.): 318-362. MIT Press.

3. T. Sejnowski and C. R. Rosenberg. 1986. "NETtalk: A Parallel Network That Learns to Read Aloud." Johns Hopkins University Technical Report JHU/EECS-86/01.

DISTRIBUTION

	Copies
Commander	
Space and Naval Warfare	
Systems Command	
Attn: Code 301	1
Code 3011	1
Code 311	1
Washington, DC 20363-5100	
Defense Advanced Research	
Projects Agency	
Attn: DSO (Yoon)	1
1400 Wilson Blvd.	
Rosyln, VA 22201	
Commander	
Naval Weapons Center	
Attn: Code 3903 (Andes)	1
China Lake, CA 93555	
Defense Technical Information	
Center	
Cameron Station	12
Alexandria, VA 22314	

Internal Distribution:

C	1
D	1
D4	1
D24 (Bailey)	1
D25	1
E231	3
E232	2
E32 (GIDEP)	1
F41 (Kruger)	1
FOX	1
G07	1
G12	1
G42 (Farsaie)	1
G71 (Gray)	1
K	1
K105	5
K10	5
K12	5
K12 (Roger)	10
K12 (Solka)	10
K13 (Shuler)	5
K14	5
K14 (Steffen)	10
K44 (Glass)	2
K52 (Farr)	2
N05	1
N06	1
N13	1
N24	1
N24 (Bailey)	1
N35 (Kuchinski)	2
N415	2
R04	1
U31	1

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1990	Final

4. TITLE AND SUBTITLE
NEURAL NETWORK IMPLEMENTATION OF AN F-14 BATTLE MANAGEMENT FUSION ALGORITHM RULE BASE

5. FUNDING NUMBERS

6. AUTHOR(S)
George Rogers, Jeffrey L. Solka, and David Steffen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Surface Warfare Center (K12)
Dahlgren, VA 22448-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
NSWC TR 90-169

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The Attack and Defense of Maritime Resources in Adverse Locals Simulator (ADMRALS) system is a distributed multiwarfare simulation system. Central to the outer air battle in ADMRALS is the F-14 battle manager, currently consisting of a rule base of 722 rules. This has been implemented using a multilayer perceptron network. A hybrid sigma-pi network has been devised for implementation on a parallel computer.

14. SUBJECT TERMS
Attack and Defense of Maritime Resources in Adverse Locales Simulator (ADMRALS) multiwarfare simulation system, outer air battle, F-14 battle manager, multilayer perceptron network

15. NUMBER OF PAGES
30

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR