Real Time Data Reduction and Analysis Using Artificial Neural Networks

Steven M. Dionisi
AFFTC
Edwards AFB, Ca 93524-8370

ABSTRACT

An artificial neural network (ANN) for use in real time data reduction and analysis will be presented. The use and advantage of hardware and software implementations of neural networks will be considered. The ability of neural networks to learn and store associations between different sets of data can be used to create custom algorithms for some of the data analysis done during missions. Once trained, the ANN can distill the signals from several sensors into a single output, such as safe/unsafe. Used on a neural chip, the trained ANN can eliminate the need for A/D conversions and multiplexing for processing of combined parameters and the massively parallel nature of the network allows the processing time to remain independent of the number of parameters. As a software routine, the advantages of using an ANN over conventional algorithms include the ease of use for engineers, and the ability to handle nonlinear, noisy and imperfect data. This paper will apply the ANN to performance data from a T-38 aircraft.

KEY WORDS

Neural Networks, Real Time Data Analysis, Massively Parallel Systems

INTRODUCTION

Small ground facilities need a process to reduce and analyze real time data that does not require dedicated programmers. One solution to the problem is the use of embedded artificial neural networks to reduce the amount of customized algorithms required for data analysis. By using specialized neural software and a spreadsheet, an engineer with limited knowledge of C language can create subroutines to duplicate complex algorithms created by dedicated programmers. The engineer can make use of the fact that once the data is entered on a spreadsheet, the neural network will learn to correlate and classify multiple data sets. Unlike a simple lookup table, the ANN will extrapolate from known data to provide solutions to previously unseen data sets.

The ANN for this paper was built on a 386 computer using Brainmaker Professional Software from California Scientific Software (CSS) and DynaMind from NeuroDynamX (NDX). Data to train the network was obtained from a T-38 flight manual. The neural network was embedded in a C routine and was run real time on a System 500 Decom from Loral Instrumentation. While limitations in time and hardware prevented the neural network to be used on a neural chip, the process for this application will be presented.

THEORY OF ARTIFICIAL NEURAL NETWORKS

The artificial neuron is based on the biological neuron. While the artificial network is not as complex as the biological system, they have several characteristics in common. Unlike digital systems that store information as discrete states in specific cells, the neural network stores information in terms of the relationship between cells. To do this, the artificial neuron consists of two stages: the weighting stage and summation/transfer stage. The stronger the relationship between neurons, the greater the connectivity or weighting from one cell to the other. A signal passing from one cell to another is multiplied by this weighting factor that may amplify or attenuate the signal. At the neuron body all inputs to the cell are summed and passed through a thresholding function. The thresholding or sigmoid function decides the signal level, zero to one, which is passed to the next neuron. [4]

Because of the sigmoid function, the artificial neuron can classify or sort several inputs. (Figure 1) This is important for applications such as safety-of-flight monitoring that require the output of several sensors to be combined and a single result (safe/unsafe) to be provided. The single neuron has limited use since it can only discriminate a very limited data set using a linear divider. The real power comes in using several neurons organized into layers. In fact, the Kolmogorov theorem states that any region, no matter how complex can be represented by a multilayer artificial neural network of no more than three layers: an input layer, a hidden and an output layer. (Figure 2)[1] The organization of neurons in this form is known as feedforward network. The ANN for this paper was a simple feedforward network with a single hidden layer. Based on the Kolmogorov theorem, a complex boundary surface will be created to delineate the safe regions of state space from unsafe regions, as in the flight envelope on a T-38 aircraft.

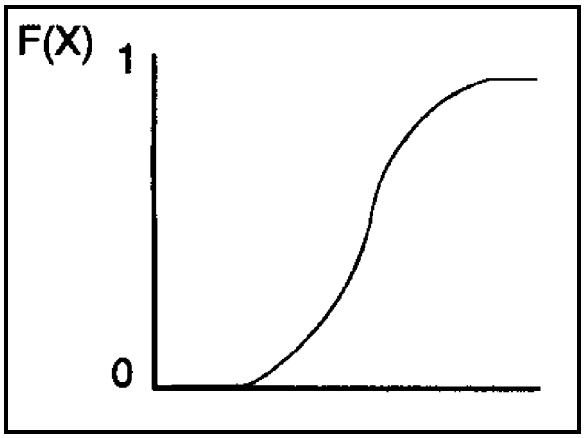

Figure 1 Transfer Function

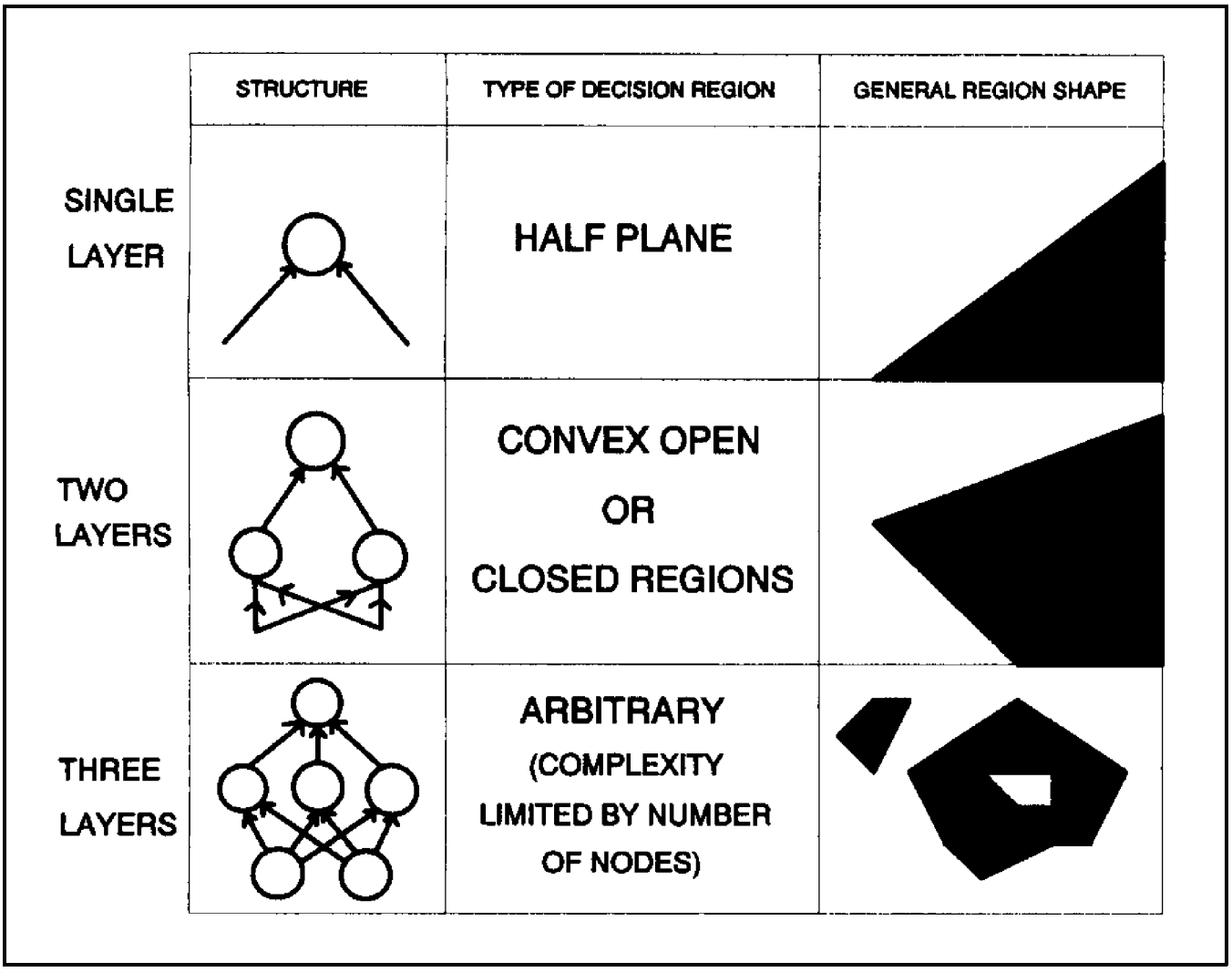

Figure 2 State Space Boundaries (Lippmen)[3]

The process of creating and using the ANN consists of three stages. The first stage consists of collecting the data and putting into a form the network can train with. The second stage is to create the network and training it. The third stage consists of setting up the ANN as software with a C language subroutine or, as hardware, with the 80170nx neural chip from Intel.

COLLECTING AND PROCESSING THE LEARNING DATA

The first step in the process was to determine what type of data would be used to train the network. While the neural network has the ability to accept several inputs and determine which variables are relevant to the output, it is the job of the engineer to limit the number of inputs, otherwise the training time will be extensive. Also, the task of data entry becomes unmanageable with a large data set. To limit the scope of the data, only two parameters from the graph of the flight envelope were used, indicated mach number and pressure altitude. (Figure 3) Additional input parameters, such as

gross weight or wing position, could have been included, if the application required them. For this example, clearly wing position is not important since the T-38 has fixed wings. The decision to remove gross weight was more difficult, for an actual test program it may be necessary. For purposes of this paper, gross weight was not necessary.

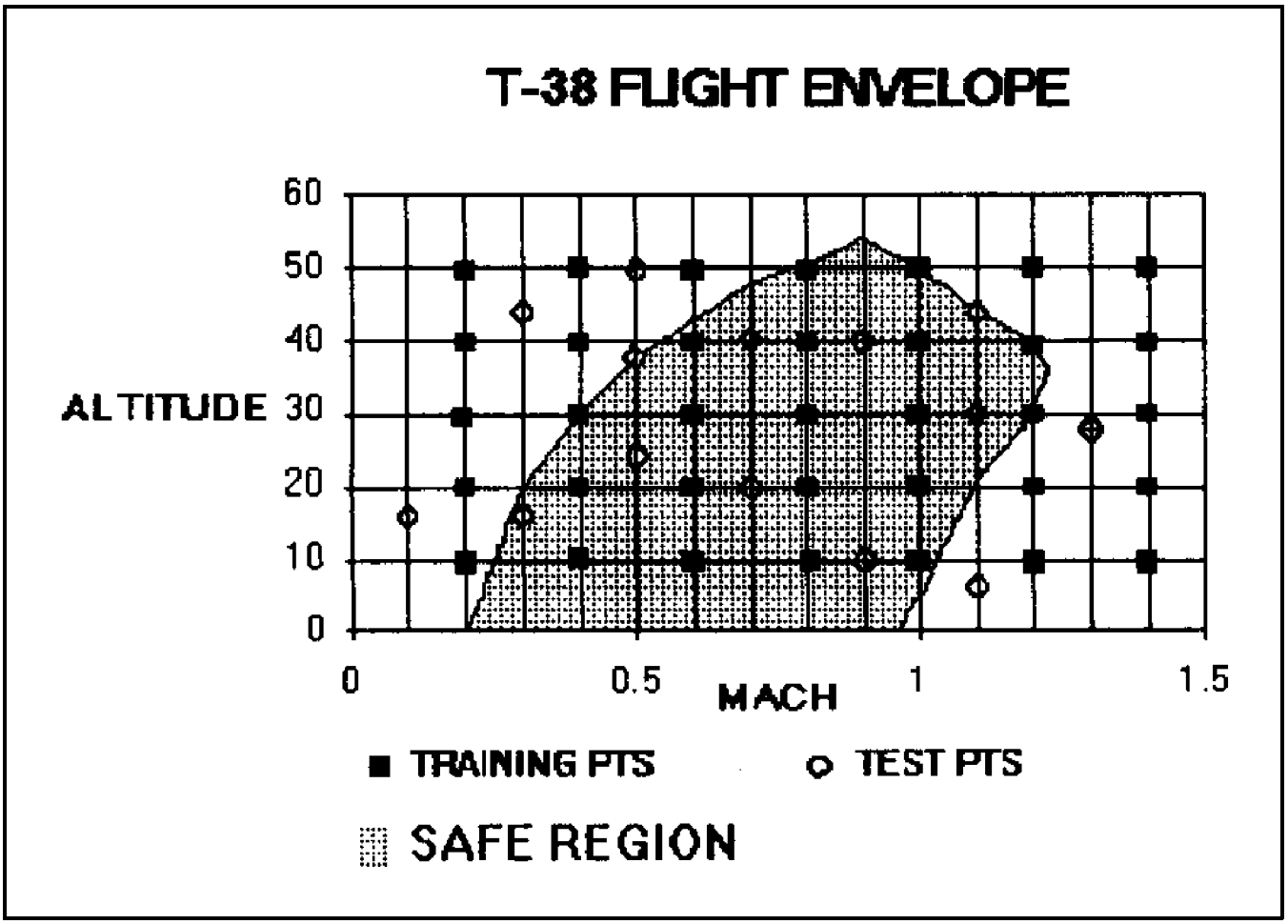

Figure 3 T-38 Flight Envelope

Besides limiting the number of inputs the engineer also needs to determine the size of the training set. The number of points is dependent on the ability of the network to train. If the neural network doesn't converge on a solution, more points may be required. Thirty-five data vectors from the graph of the flight envelope (standard day) were entered onto a spreadsheet. The training set contained vectors consisting of three parameters < mach, altitude, and unsafe/safe >. For unsafe/safe, a "0" was inserted for unsafe, "1" for safe, and any points on the line were weighted at .5.

If any of the parameters vary greatly, such as altitude (0-60,000 ft), those parameters need to be transformed/normalized. Transformation/normalization is the process of manipulating the raw data so that it falls with in a set range of values. Because the ANN training algorithms vary the weighting in small incremental steps, the larger the

range on the input or output the longer the training will take, if it converges at all. Ideally the inputs/outputs should vary from 0 to 1 to optimize the training time. For nonlinear data, a transformation function is used such as l/x or log x, otherwise for linear data the scaling is done by normalization. The equation for normalizing linear data is as follows:

$$X_N = \left[\frac{X_D - d_{MIN}}{d_{MAX} - d_{MIN}} * (TF_{MAX} - TF_{MIN}) \right] + TF_{MIN} \qquad \text{(Eq. 1)}$$

X_N = Transformed data point
X_D = Raw data point
d_{min} = Minimum value of parameter
d_{max}= Maximum value of parameter\
TF_{max}= Maximum value of transfer/sigmoid function
TF_{min}= Minimum value of transfer/sigmoid function

For the case of altitude: d_{max} = 60,000, d_{min} = 0, TF_{max} = 1, TF_{max}=0. Setting the transfer function to something other than 0 to 1 is possible if the maximum output of the net is required to be greater than one, but then the training becomes more complex. For this application, TF_{max} = 1 where the maximum output of the net (safe) is equal to 1. The input mach, unlike altitude, only varies from 0 to 1.6 and no normalization is necessary because it is close enough to the ideal range for an input.

TRAINING THE ANN

The second step in the process is constructing and training the network. To do this, the spreadsheet is imported, after data processing, into the neural software for construction and training of the ANN. Both neural software packages will automatically generate the net size based on the imported data. The software also allows the user to adjust the network size if more neurons are required. For example, if the net is not converging, more neurons may be added to the hidden layer. For the majority of situations, the generated nets will work. In this example, the net consisted of two input neurons, 20 neurons on the hidden layer, and one output neuron.

The net is initially generated with randomized weighting between layers, therefore training is necessary before the net is useful. The training is accomplished by a method known as backpropagation. This method applies individual sets of data (one row on the spreadsheet) to the network and compares the actual outputs from on the spreadsheet to the predicted output from the neural net. The difference or error is propagated back through the network and the weighting of each branch is adjusted to

minimize this error. This process is repeated several times until all sets of inputs generate outputs that meet the established tolerance for error as chosen by the user.

Several factors may require changing if the net cannot meet the chosen tolerance. The following factors prevent the ANN from converging:

Too small a sample size
More neurons are required in the hidden layer
The conflicts in or the order of the data may cause the net to unlearn previous facts
The learning steps (speed of training) may require changing

It is a process of trial and error to select the right factors to change.
Theoretically, there is no guarantee the neural net will be able to converge on the correct outputs regardless of the tolerance. In the majority of cases, these problems will not occur. For more information on the symptoms of nonconvergence and their causes, refer to the book by Lawrence in the reference . [2]

After the network converges, the test data is passed through the neural net. This is data the net did not train on. Normally 10% of the training data is set aside for testing. Several software packages can do this automatically while preparing the spreadsheet for training or the user may create a separate spreadsheet just for test data. For this example, a separate spreadsheet was created for test data. The results for this net show 85 percent accuracy in predicting the correct output for the test data. The accuracy was calculated by dividing the safety factor into two areas: unsafe(0-.49), safe(.50-1). A good prediction was one where the actual and predicted were in the same region. The ANN is ready for use, if it meets the necessary accuracy required by the engineer. (Figure 4)

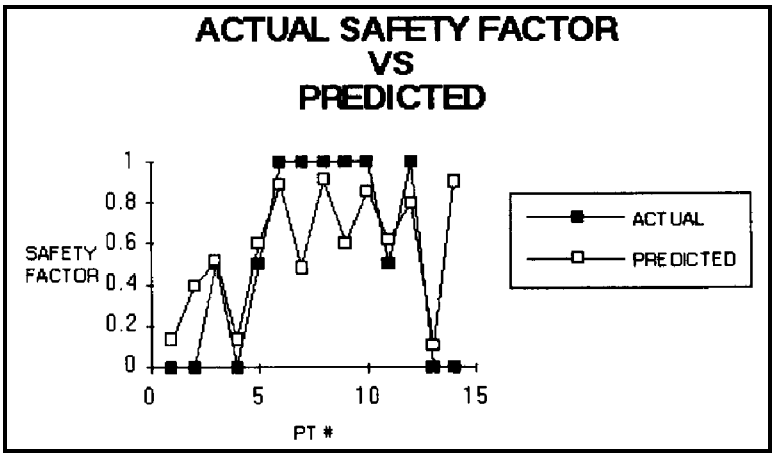

Figure 4 Predicted Using Net 1

To improve the accuracy of this system, several factors could be changed. The training tolerance could be lowered and the training time increased. The flight envelope could be graduated in smaller increments besides 0, .5 and 1, or additional outputs could be added to the net. Instead of one output for unsafe/safe, two outputs for unsafe and safe or three for unsafe, marginal and unsafe could be added.

To improve this system, three outputs were created and the flight envelope was graduated. Unsafe was defined by expanding the envelope by 10% and points outside this were given a value of one. Points between the new boundary and the old were uniformly graduated from one at the new boundary to zero at the old. (Figure 5) The same process was used to create the safe region once a boundary was established by constricting the envelope by 10%. Marginal was defined as zero at the new boundaries and one at the old. Any points between were uniformly graduated. The accuracy increased to 93%, 8% over the previous method.

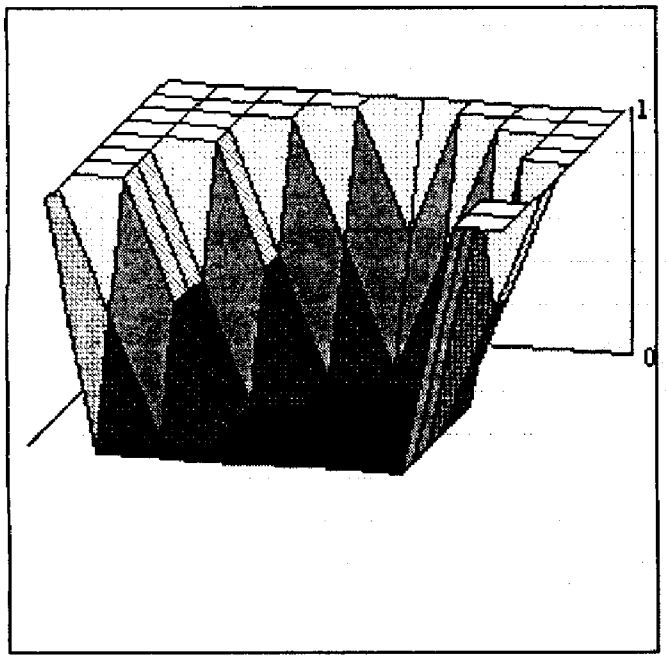

Figure 5 3D view of region

Unfortunately, 93% was still not good enough. Test point #14 was still in the wrong region. The ability of the ANN to discover relationships between the input data and the outputs can cause problems for the engineer, if the net finds a correlation he was not aware of. In this case, the net determined that test point #14 was a safe region, in contradiction with the training set.

Referring back to the original graph of T-38 flight envelope in the flight manual, test point #14 was within a safe region for an aircraft if it was in a 60° dive at high mach. To ensure the net learned the proper relationship, the training set was reinforced with additional data points in the area of test point #14. After testing with the new training set, the ANN achieved 100% accuracy.

By creating three outputs, more information can be extracted real time, such as trends can be established. The first network stated the current situation of the aircraft, which was whether it was in a safe region or not. The current network coupled with probablistic logic, such as fuzzy logic, also supplies information on the trend of the aircraft toward a safe or unsafe region. For example, an altitude of 20 KFT and a mach of .4 equates approximately to the following output vector: Safe = .6, Marginal = .3, and Unsafe = 0. As the mach decreases, the ratio changes, Safe decreases to .45 and Marginal increases to .55. (Figure 6) From this data, it can be determined that the aircraft is moving toward unsafe state; with this information corrections can be made by the pilot before he reaches an unsafe position.

IMPLEMENTING THE ANN

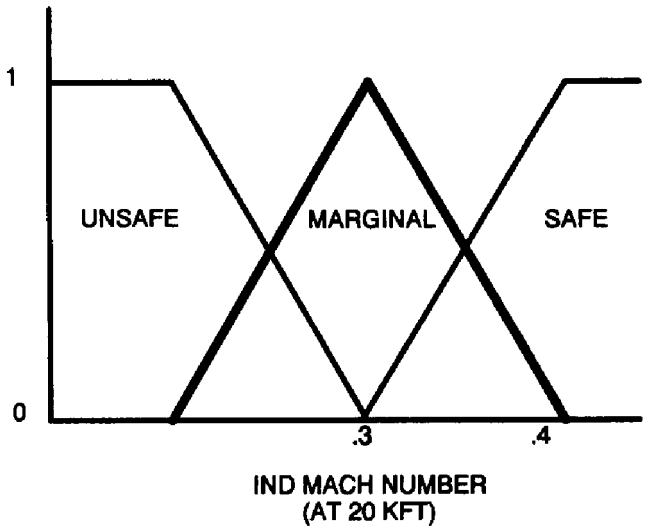

Figure 6 Crosscut of 3D views

The third step is to implement the ANN into the application. Either a C language subroutine can be created or the network can be loaded directly into a neural chip. Both software packages include C subroutines that accept the ANN. There are slight differences in the approach of each package in embedding the ANN. CSS provides the C source routine and is transportable across platforms. NDX uses libraries complied exclusively for a PC (80X86) based system. Because of this difference, only the CSS routine was run directly on the System 500.

CSS includes a program named Runtime.c to run as a feedforward network to link with applications. To use this program, three of the default filenames called by Runtime must be modified. The first is BrainRTS.net which contains the trained network. The training program Brainmaker creates this file from the training sets, but the filename is user defined. So the default must be renamed to the user defined name, in this case Fltenv.net. The other programs, BrainRTS.in and BrainRTS.out, are files containing the data to be processed and the resultant output. For this project, they were renamed Altairsp.in and Safety.out. The C code was loaded into the System 500. At this point the network is identical to any other algorithm used by the decommutator.

The NDX routines can be used by PC decoms or possibly a System 500 linked to a PC. To use the ANN, created by Dynamind, in a C application requires adding neurolnk.obj and neurolnk.lib to the application project file. Also the modules that call the routines need the following statement: # include < neurolnk.h > . By using the Neurolink from NeuroDynamX, the application is linked to the training program Dynamind. Once installed with Neurolnk software, the retraining of the ANN can be accomplished without recompiling the application.

By using external hardware, a PC based programmer from Intel, the ANN can be loaded directly into the 80170nx. Embedding the ANN directly into a neural chip offers several advantages, such as reduced size and increased processing. For the case where several transducers are sampled and digitally processed to form a combined product, a neural chip simplifies the process. The 80170NX can accept 64 analog inputs with varying voltages -2.5 to +2.5 volts. The neural network' s ability to handle noisy analog signals avoids the cost and size involved with filtering, A/D conversion

and digital processing. Due to the massively parallel nature of neural networks, neural chips achieve very high performance levels. The 80170NX can perform more than two billion connections per second. The typical 12 bit A/D performs 50,000 connections per second. For applications on aircraft, where processing, memory and space are at a premium, the neural chip could be used to replace certain digital systems that provide information directly to the pilot.

CONCLUSIONS

The artificial neural network can be used to create custom algorithms for use in real time data reduction and analysis. By using the ability of the ANN to correlate data sets, the engineer can find a fast solution to atypical problems. While there are other methods to solve non-linear functions with noisy inputs, the ANN provides one of the easiest and fastest solutions. For the small facility, the artificial neural network provides the means to maximize hardware productivity, without increasing manpower, by providing software capabilities previously unavailable.

ACKNOWLEDGEMENTS

I would like to thank my wife, Maggie, for her help and support in preparing this manuscript. Also, I would like to thank Curtis Hopper and Charlie Perry for their help with the System 500.

REFERENCES

1. Aleksander, I. and H. Morton. An Introduction Neural to Computing. New York: Chapman and Hall, 1990.

2. Lawrence, J. Introduction to Neural Networks and Expert Systems. Nevada City, Ca: California Scientific Software , 1992

3. Lippman, R.P. "An Introduction to Computing with Neural Nets." IEEE ASSP Magazine, April:4-22, 1987

4. Minsky, M . and S. Papert. Perceptrons: An Introduction to Computational Geometry. Cambridge, Mass: MIT Press, 1969.