Fermi National Accelerator Laboratory

FERMILAB-Conf-90/20

Neural Networks for Triggering*

B. Denby
Fermi National Accelerator Laboratory
P.O. Box 500
Batavia, Illinois 60510

M. Campbell
University of Michigan
Ann Arbor, Michigan 48104

F. Bedeschi
INFN Sezione di Pisa
Pisa, Italy

N. Chriss and C. Bowers
University of Chicago
Chicago, Illinois 60637

F. Nesti
Scuola Normale Superiore
Pisa, Italy

January 1990

* Invited talk presented by B. Denby at the 1989 IEEE Nuclear Science Symposium, San Francisco, January 15-19, 1990.

Operated by Universities Research Association Inc. under contract with the United States Department of Energy

Neural Networks for Triggering

B. Denby, *Fermi National Accelerator Laboratory* †
M. Campbell, *University of Michigan*
F. Bedeschi, *INFN Sezione di Pisa, Italy*
N. Chriss, C. Bowers, *University of Chicago*
F. Nesti, *Scuola Normale Superiore, Pisa, Italy*

Abstract

Two types of neural network beauty trigger architectures, based on identification of electrons in jets and recognition of secondary vertices, have been simulated in the environment of the Fermilab CDF experiment. The efficiencies for B's and rejection of background obtained are encouraging. If hardware tests are successful, the electron identification architecture will be tested in the 1991 run of CDF.

1 Introduction

A trigger in a high energy physics experiment is a device which decides whether an event is 'interesting' or 'not-interesting' based upon the configuration of the various pieces of data which make up the event. The trigger thus can be considered as implementing a binary function of many variables. If the function has value '1', the event is accepted, and if it has value '0', the event is rejected.

Level-1 triggers based upon rather simple linear functions, such as total energy, total transverse energy, and missing transverse momentum, provide large rejection factors. However, in most experiments it is also necessary to provide two additional levels of trigger which implement the more sophisticated pattern recognition algorithms necessary to bring rates down to levels suitable for storage on recording media. Since levels-2 and 3 are normally implemented with some combination of special purpose hardware and arrays of programmable processors, it may seem strange to think of the trigger here as a 'device which implements a binary function of many variables', but let us retain this definition. The function implemented can be extremely complicated.

2 Beauty Triggers

Let us consider the example of beauty triggers. The two most promising methods of triggering on beauty are detection of an electron (or muon) from the semileptonic decay of a beauty meson and the detection of secondary vertices from the beauty decay. In the first case the pattern recognition task is that of identifying an electron shower which will in general be contained within a jet. The input variables are the energy deposits in a calorimeter, and, based on these, the trigger implements a function whose value is '1' if an electron is present and '0' if no electron is found. In the second case, the trigger operates upon the locations of hits in a vertex detector, finds tracks, and determines whether these tracks emanate from one or multiple vertices. The trigger function has value '0' for a single vertex, and '1' for multiple vertices. The complexity of the trigger function in these two cases can be appreciated by noting that of the order of several hundred lines of code would be necessary to implement them on a computer.

3 Developing Trigger Algorithms

Trigger algorithms are usually developed with the aid of Monte-Carlo data sets, which contain 'interesting' (signal) events and 'not-interesting' (background) events. There are typically a number of parameters in the algorithm whose optimum values are learned by passing repeatedly through the Monte-Carlo 'training set' and varying their values so as to achieve maximum acceptance of signal and rejection of background. An iterative method of learning the optimum parameter values is necessary since there is no straightforward way of calculating them.

This is the traditional method of developing trigger algorithms. A novel method would be to attempt to determine the form of the function which the trigger implements and simply *calculate* the value of this function for every event. An arbitrary function of many variables can be approximated with good accuracy by representing it as a linear combination of known functions and then ascertaining the coefficients of these functions in the linear combination. One example is to represent the function as a Fourier series. Another would be to use the program MUDIFI [1] developed at CERN. This program allows the user to approximate an arbitrary function by a linear combination of polynomials in the input variables. The coefficients in the linear combination, the number of polynomials, and the degrees of the polynomials are parameters which are determined by an iterative fitting procedure. The program

† Fermilab is operated by Universities Research Association, Inc., under contract with the U.S. Department of Energy.

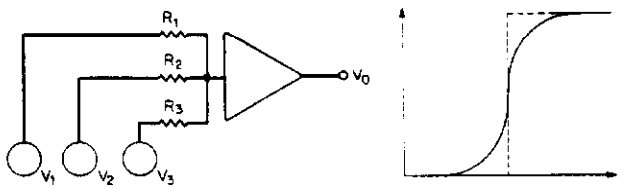

Figure 1: A weighted sum of the V_i appears at the input to this 'neuron', which is just an amplifier with a 'sigmoid' response.

effectively constructs the desired function out of known polynomials by constraining it to give the correct answer for each of a set of 'training events' consisting of the values of the input variables and the desired function value. This use of training samples to approximate unknown probability distributions is a standard technique in pattern recognition [2]. Although it is 'novel' to attempt to develop a trigger function in this way, is there any advantage to doing so?

4 Neural Networks

Using a standard computer to evaluate a function which mimics an algorithm is probably not advantageous over simply running the algorithm on that computer. The advantage of the function evaluation lies in that it may be possible to build a device which evaluates the function electronically, and, hence, very much more quickly than the algorithm running on a standard computer. Electronic generation of simple functions is certainly feasible, but how can we generate the very complicated functions required for experimental triggers? The answer, oddly enough, comes from biology. Animal nervous systems construct sophisticated pattern recognition circuits from networks of many simple processing elements called neurons. In the past few years enormous advances have been made in the field of artificial neural networks which seek to imitate the style of computing architecture found in animal nervous systems.

Figure 1 shows the basic element of a neural network, i.e., a neuron. Each neuron is an analog device which sums signals from many other neurons (in this case, the three V_i in the figure) at its input. Associated with each pair of neurons is a weight, encoded in the resistors shown in the figure, which multiplies the sender's signal before it passes to the receiver. In most models, neurons have a 'sigmoid' response function which is simply a 'rounded-off' step function.

Figure 1 is also familiar from high energy physics. It is just the diagram of a discriminator circuit. We could treat a set of calorimeter cells as 'input' neurons, encoding the energies they contained by voltages. These voltages are transformed into currents in passing through the resistors. If we make the resistor values proportional to one over the cosine of the polar angle of the cell, the sum formed at the neuron/discriminator input will be just the summed transverse energy in the calorimeter. We could choose the amplifier gain such that saturation (i.e., output = '1') occurs for transverse energies above some threshold. This 'calculation' of $\sum E_T$ is very fast since it takes only as long as it takes the signals to propagate through the circuit. Alternatively, we could use paddle counters in a beam line telescope as the input neurons. In this case all the resistors would have the same value, and we can arrange that saturation occurs when all paddles have been hit. These kinds of techniques have been used for decades in constructing fast trigger logic.

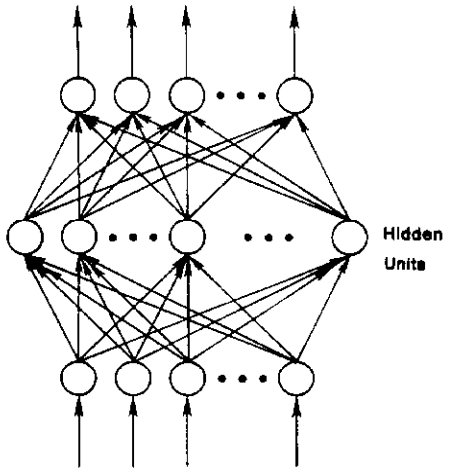

Figure 2: Generic 3-layer feed-forward network of neurons (circles). Weights reside on the lines connecting neurons.

These are examples of rather simple functions implemented by our 'neural network'. Much more complicated functions can be implemented by using many more neurons interconnected in complicated ways. In the field of artificial neural networks, it is traditional to consider two basic types of architectures, the feed forward network and the recurrent network. We turn now to an example of each of these types on network drawn from current research on neural networks for high energy physics applications. Both examples deal with beauty triggers.

5 Calorimetry Based B Trigger

5.1 Feed-Forward Nets

The basic feed forward architecture is shown in figure 2. The first layer is called the input layer, the second layer the 'hidden' layer, and the third layer the output layer.

Normally there is also an extra 'bias' neuron with a constant value of 1 which connects via weights to all hidden and output units. The resistive connection weights are understood to reside on the lines in the figure connecting the units together.

It has been shown [3] that any well behaved function mapping n real variables to m real variables can be approximated to an arbitrary degree of accuracy by a feed-forward network of sigmoid units comprised of n input units, m output units, and a single layer of hidden units. The neural network approximates arbitrary functions by using many shifted, scaled sigmoids to build up the desired function piece by piece. The shifting is provided by the bias units, and the scaling is provided by the multiplicative weights. Thus, the action of the network behaves in a way qualitatively similar to a Fourier series expansion or an expansion in polynomials as in MUDIFI, except that the expansion is in terms of sigmoids. The sigmoid is a good choice for an expansion which is to be realised with hardware components since all the active elements (neurons) can be the same, and need only be accurate over a relatively small range of inputs. Designing hardware components which represent an ensemble of polynomials accurately over their entire range would be considerably more difficult.

5.2 Trigger Simulation

The ISAJET [4] Monte-Carlo was used to generate events of the form $p\bar{p} \rightarrow b\bar{b} + X$ where one of the b's decays semileptonically, i.e., $b \rightarrow e\nu c$. A sample of real minimum bias events from the most recent run of the CDF [5] experiment was used as the background sample. The effective tower size of the signals used in the simulation was $\Delta\eta = 0.2$ by $\Delta\phi = 15$ degrees. A training file of evenly mixed signal and background events (about 150 signal and 300 background) was created from the ISAJET data and the minimum bias data after applying identical cuts on total E_T and the size of the largest electromagnetic tower to signal and background samples. For the results presented here, the cut used was: $\sum E_T \geq 18.$ GeV *or* largest electromagnetic tower $\geq 4.$ GeV. This cut reduced the background by about a factor of 100 and the signal by about a factor of 2. The task was to identify, in the training file, those events which contained B's by identifying energy clusters which contained an electron.

The network used in the B trigger simulation had 128 input units and 2 output units. The number of hidden units was varied between 50 and 125 in steps of 25 to see the effect of this on trigger performance. The input quantities were the energies contained in an 8 by 8 cell region of interest taken from the calorimeter. The region of interest was selected by finding the cell in the calorimeter with the largest electromagnetic energy deposit and taking an 8 by 8 cell block around this cell. A typical event before selecting the region of interest is shown in figure 3. The two output units encode whether the event is signal, i.e., (0 1), or background, (1 0).

The network was trained using a standard training algorithm called 'back-propagation' [6]. Backpropagation performs gradient descent with respect to the weights on an energy function that measures the deviation, summed over the training set, of the network response from the desired response.

After training, the network was tested on a new sample containing signal and background events (about 50 of each) not contained in the training sample. The network correctly identified 65% of the B-jets, while rejecting 95% of the background. This result was independent of the number of hidden units within the statistics of the samples used. When tested on the *training* sample, the network identified 95% of the B-jets correctly, which may indicate that the network established a rule for identifying 65% of the events and simply 'memorised' the remaining 30%. This points out the importance of testing the network on an independent data sample.

These results, though promising, are preliminary. Work is continuing to improve the acceptance and rejection of the trigger. Once the optimum parameters of the network are determined, it is intended to build the network as a hardware device and test it during the next run of CDF in 1991. The input to the network will be the analog levels provided by the existing cluster finder in the CDF trigger.

The trigger decision time for this circuit depends only upon the propagation time of signals from the input layer to the output layer. As only passive components and simple amplifiers are involved, this can be expected to be of the order of several hundred nanoseconds. A few prototype neural network chips have already been produced in research labs and in industry. Propagation times for these chips, which have not been designed with the needs of high energy physics in mind, are nonetheless of order one microsecond [7]. Thus is appears as though the neural network trigger will provide a way of significantly enhancing the data sample with beauty events while adding only an insignificant amount to the trigger decision time.

6 Secondary Vertex Trigger

6.1 Recurrent Nets

The basic recurrent network architecture is shown in figure 4. A single layer of neurons serves as input and output. The connection strengths are usually taken to be symmetric, i.e., the weight from neuron i to neuron j is the same as that from j to i. With this prescription, it can be shown [8] that the activations, a_i, approach a state which minimizes an energy function $E = -\frac{1}{2}\sum_{i,j} w_{ij}a_i a_j$ where w_{ij} is the weight between neurons i and j. Initial values are placed on the input lines, and the final values appear there after the the circuit settles. Although back propagation algorithms exist for recurrent networks, they are complicated by the fact that signals may pass several times through the network before settling. It is more common to use intuition to devise appropriate weights for recurrent networks.

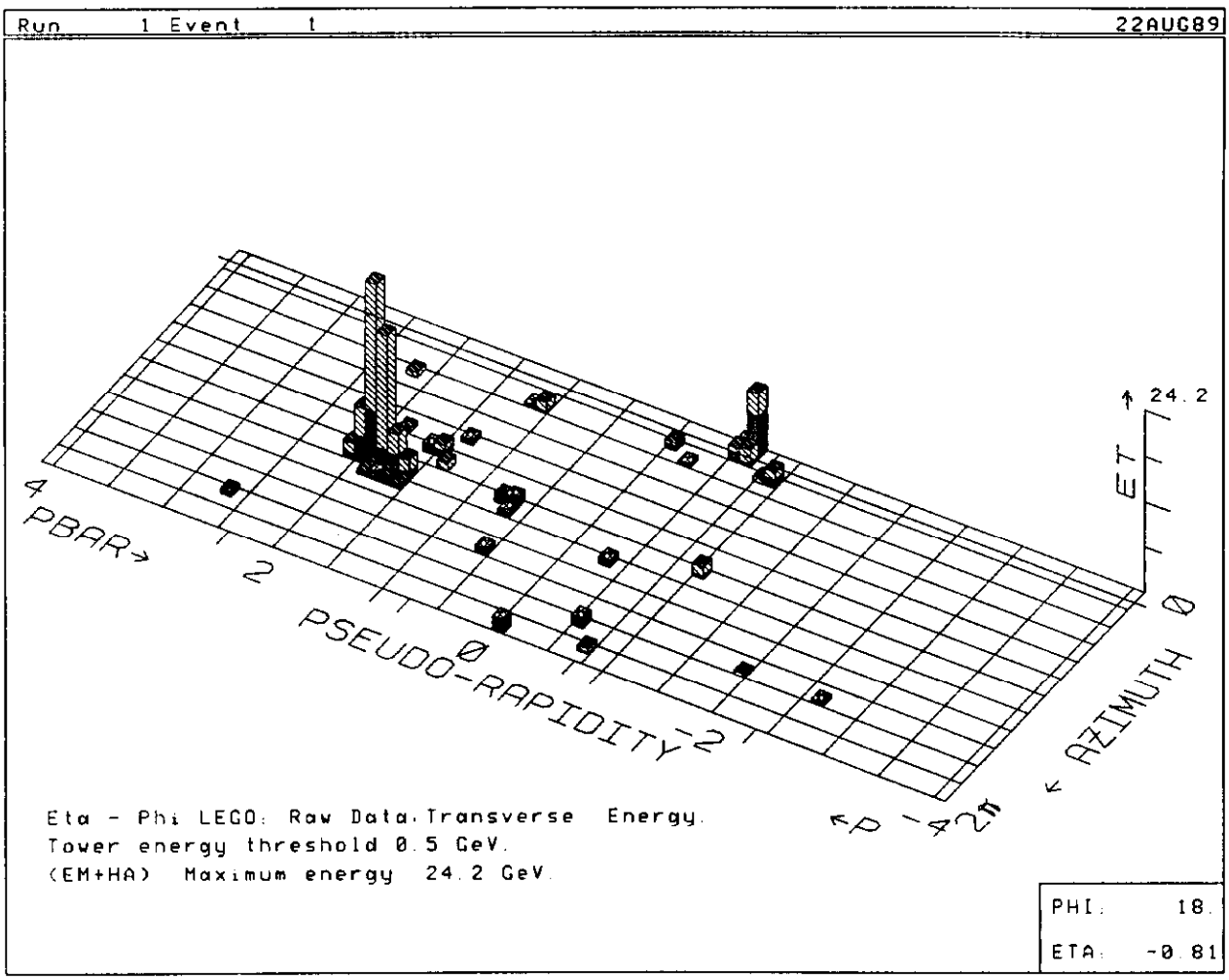

Figure 3: 'Lego' plot of simulated $b\bar{b}$ event in CDF calorimeter, with one b decaying semileptonically.

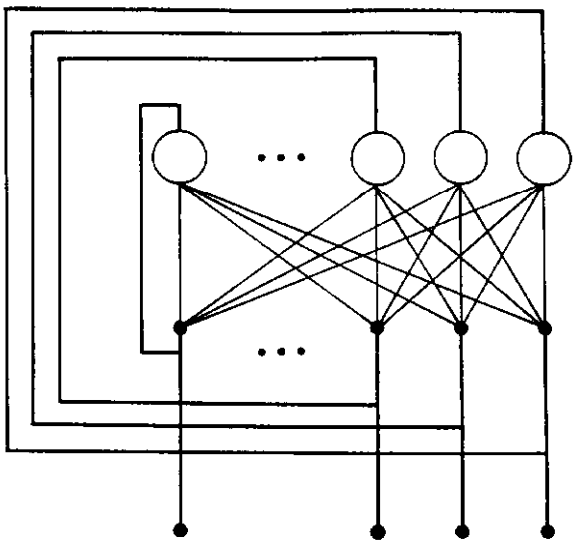

Figure 4: Recurrent neural network. Initial values are put at the inputs (lower large dots), which are tied to the sigmoid outputs, and final values appear there after circuit settles. Weights reside on lines connecting upper large dots to neurons (circles).

6.2 Trigger Simulation

The ISAJET Monte-Carlo was again used to generate $b\bar{b}$ events and a sample of 2 gluon jet background events of comparable p_t. The B particles are allowed to decay into their secondary decay products. The geometry used was that of the SVX detector to be installed in the CDF experiment. This is a $r-\phi$ tracking device consisting of four layers of 60μ pitch silicon microstrip detectors arranged in a cylindrical geometry around the beam pipe.

We assume the presence of associative memories which give fast track parameters for tracks in the SVX. Such memories are currently being fabricated at INFN Pisa [9]. We also assume a fast track processor based on the outer tracking chamber which provides p_t information for tracks down to .5 GeV/c. It is necessary to have this p_t information in order to retain sufficient accuracy in projecting tracks back to the region of the primary vertex. Without it, it is necessary to put a cut at about 3 GeV/c on tracks, which effectively annihilates the B signal.

The tracks generated by the Monte-Carlo are approximated by tangent straight lines in the vicinity of the origin, and parametrized by D, the distance of closest approach to the origin, and ϕ, the azimuthal angle of the track. The calculated parameters are then smeared with smearing functions based upon the known properties of the CDF central tracking chamber and the expected properties of the SVX. It can be shown that in this parametrization, tracks from the primary vertex will lie on a horizontal line in $D-\phi$ space, and tracks from secondary vertices will lie on lines at some angle to horizontal (the angle is proportional to the distance from the origin of the vertex). An example is shown in figure 5a. The size of the beam spot is small compared with the D resolution in this plot. This means that we can remove all primary tracks by simply cutting around $D=0$. The actual cut used required the distance of the track from $D=0$ to be more than 3 times the error on D, which is a function only of p_t. (A straight cut on D of 175 microns only slightly worsens the performance of the trigger.)

Bottom Jets:

$30 < p_t(GeV) < 100$	efficiency for B's:	70%
	background accepted:	0.26%
$50 < p_t(GeV) < 100$	efficiency for B's:	81%

Top Jets:

$M_{top} = 100\ GeV$	efficiency for B's	50%
$M_{top} = 120\ GeV$	efficiency for B's	68%
$M_{top} = 150\ GeV$	efficiency for B's	74%

Table 1: Recurrent neural network trigger efficiency and rejection figures for B events. Background sample contained 1500 events.

After deletion of all primary tracks, a recurrent network was used to try to find the remaining lines. In the neural network approach, which is similar to one used earlier for track finding [10], pairs of tracks define a link which is identified with a neuron. The neurons reinforce each other to the degree that the angles of their links are similar, i.e., $w_{ij} \sim exp(-A\theta_{ij}^2 - B\Delta\phi^2)$, where w_{ij} is the weight between neurons i and j, A and B are positive constants, θ_{ij} is the absolute angle in $D-\phi$ space between links i and j, and $\Delta\phi$ is the difference between the ϕ intercepts ($D=0$) of the two neurons. There is also a leakage term which causes the neurons' activations to decay to zero in the absence of reinforcement from other neurons. In the initial state, the neurons for all possible links are activated. As the network evolves only those links which have neighbors of similar orientation will remain activated. In figure 5, the network has found secondary vertices coming from both B's in the event (neurons indicated by links between tracks).

The trigger requirement is at least 2 neurons on at end of evolution. The efficiency and rejection of this trigger are summarized in table 1. Also presented are the results for B's coming from top jets which were produced in separate runs of ISAJET for three values of the top quark mass. The figures presented are based upon a fixed interaction point at the center of the interaction region. Allowing a more realistic 35 cm. r.m.s. spread in the z position of the interaction point reduces the trigger acceptance by about a factor of 2., both for B and top jets. The background acceptance figure, based on 1500 events, excludes background from charm, which amounted to an additional 1.1% (this figure is given for completeness, it should not be used, to deduce the charm to strange ratio in the sample,

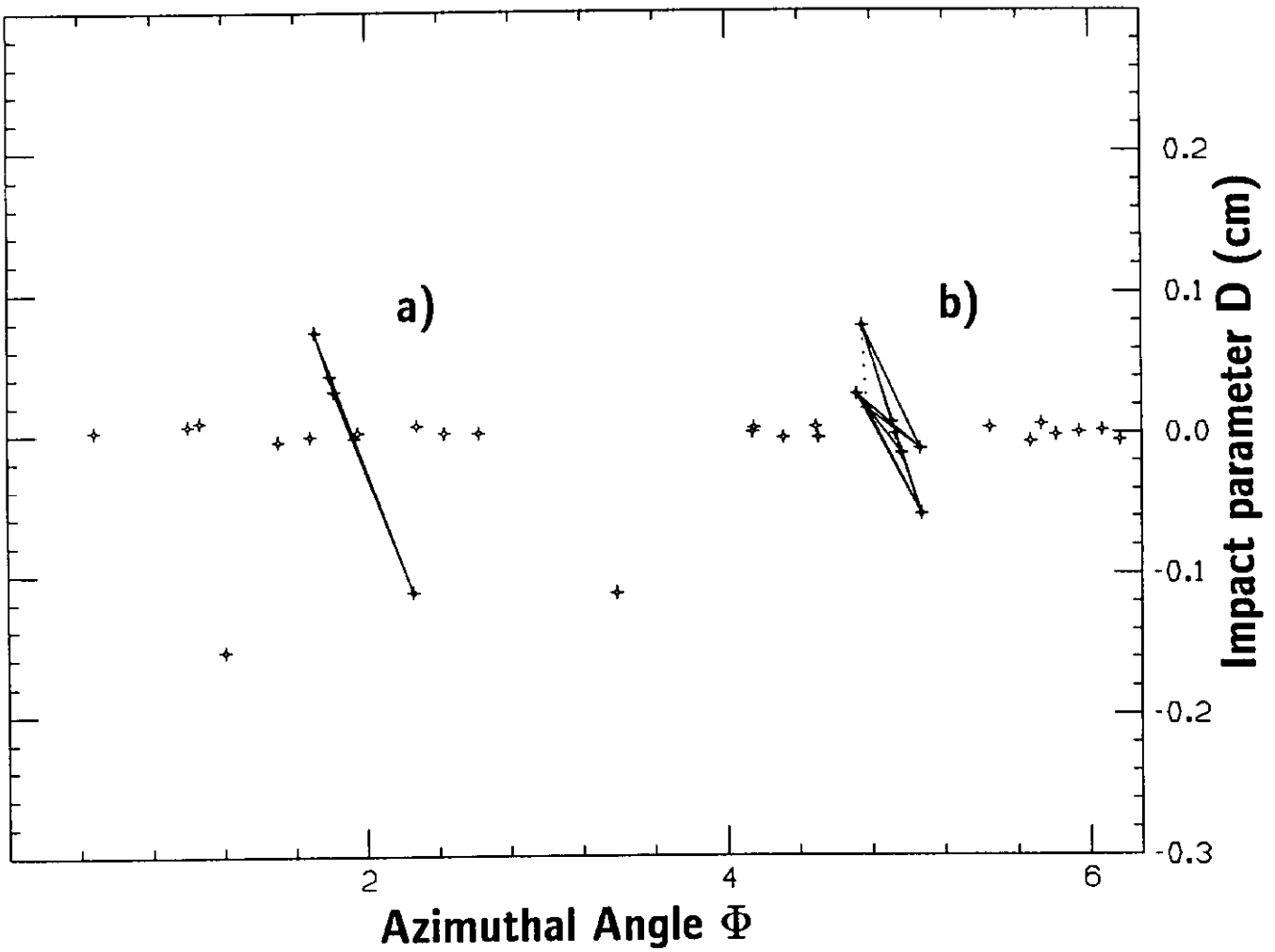

Figure 5: Tracks appear as points in $D - \phi$ space. Primary tracks lie near $D = 0$, while tracks from secondary vertices lie on angled lines. Solid lines are 'valid' neurons, dotted lines are neurons which faded away. a) and b) are explained in the text.

for instance).

In B decays, there often are at least two secondary vertices, one from the B, and one from a charmed particle produced in the decay of the B. In these cases, the tracks will lie on two separate lines in $D - \phi$ space. Because of relatively low multiplicities of the decays and inefficiencies, the resultant can look more like a cluster of nearby points than two lines (figure 5b). Nevertheless the network distinguishes such events well from background events. The reason is that in the background events, although there can be a sizeable number of high impact parameter tracks, they seldom form either lines or clusters, so that all neurons (links) formed fade away. For the signal events, whether in the form of a line or a cluster, there are normally enough links of similar orientation present that reinforcement occurs.

The efficiency and background rejection figures look extremely promising. It is clear that a secondary vertex trigger based upon this technique, if it can be realised, will be an extremely valuable tool. As in the case of the feed forward network, a hardware implementation of the neural net trigger should be extremely fast since it is only necessary to allow the circuit to settle. As mentioned earlier, prototype circuits have settling times of order 1 microsecond [7]. In addition to the neural net, it will be necessary to have some sort of preprocessor which calculates the neuron parameters and connection strengths for each event. Although we do not discuss here the architecture of this device, the calculations it does are straightforward and parallel, so that it should not present any major construction challenges nor increase significantly the trigger decision time.

7 Conclusion

Recent developments in research on artificial neural networks have created a renewed interest in computing architectures which mimic animal nervous systems and brains. Neural architectures can accurately approximate arbitrarily complex functions, including the kinds of functions implemented by trigger systems in high energy physics experiments. Large scale neural networks implemented in silicon are beginning to appear, and the small settling times of these circuits, of order 1 microsecond, make them highly appropriate for incorporation into trigger systems. We have used examples of the two most popular neural architectures, feed-forward and recurrent, to illustrate how neural networks can be applied to beauty triggers. Work is continuing on these two applications with the intent to install them for testing in upcoming runs of the CDF experiment at Fermilab.

References

[1] R. Brun, M. Hansroul, H. Wind, *MUDIFI Multidimensional Fit Program*, DD/EE/80-1, Data Handling Division, CERN, 1211 Geneva 23, Switzerland.

[2] Harry C. Andrews, *Introduction to Mathematical Techniques in Pattern Recognition*, New York, Wiley Interscience, 1972.

[3] Robert Hecht-Nielsen, *Theory of the Backpropagation Neural Network*, proceedings of the International Joint Conference on Neural Networks, volume I, pp. 593-605, Washington, D.C., 18-22 June, 1989, IEEE Catalog no. 89CH2765-6.

[4] F. Paige, S.D. Protopopescu, ISAJET Monte Carlo, BNL 38034(1986), Brookhaven National Laboratory.

[5] F. Abe et al., (CDF Collaboration), *Nucl. Inst. Meth* **A271**, (1988) p. 387.

[6] David E. Rumelhart, James L. McClelland, et al., *Parallel Distributed Processing, Explorations in the Microstructure of Cognition*, MIT Press, 1986, volume 1: *Foundations*, chapter 8:*Learning Internal Representations by Error Propagation.*

[7] M. Holler, S. Tam, H. Castro, R. Benson, *An Electronically Trainable Artificial Neural Network (ETANN) with 10240 "Floating Gate" Synapses*, proceedings of the International Joint Conference on Neural Networks, volume II, pp. 191-196, Washington, D.C., 18-22 June, 1989, IEEE Catalog no. 89CH2765-6.

[8] J.J. Hopfield and D.W. Tank, *Biological Cybernetics* **52** (1985) p. 141.

[9] M. Dell'Orso and L. Ristori, *VLSI Structures for Track Finding*, proceedings of the International Conference on the Impact of Digital Microelectronics and Microprocessors on Particle Physics, Trieste, Italy, 28-30 March 1988, published by World Scientific Pub. Co. ISBN 9971-50-742-0.

[10] B. Denby, *Computer Physics Communications* **49** (1988) pp. 429-448.