ALEPH 91-62
SOFTWR 91-01
G. Stimpfl
13.5.1991

# Recognition of Decays of Charged Tracks with Neural Network Techniques

**Georg Stimpfl-Abele**
Laboratoire de Physique Corpusculaire
Université Blaise Pascal
Clermont-Ferrand
F-63177 Aubiere, France

May 1991

**Abstract**

We developped neural network learning techniques for the recognition of decays of charged tracks using a feed-forward network with error back-propagation. Two completely different methods are described in detail and their efficiencies for several NN architectures are compared with conventional methods. Excellent results are obtained.

*(Submitted to Computer Physics Communications)*

## 1 Introduction

The recognition of particle decays belongs to the large area of discrimination and classification tasks in experimental High Energy Physics where one has to choose between different hypotheses. This can be done by applying cuts on the variables with the highest discrimination power or using special methods like discriminant multivariate analysis or hierarchical classification. An other approach are neural network (NN) techniques.

We use a NN learning algorithm with error back-propagation on a feed-forward network for the discrimination between decaying and non-decaying tracks. The behaviour and performance of this method is studied in detail for the decays $\pi^{\pm} \rightarrow \mu^{\pm} + \nu$ at three pion energies ($E_{\pi}$ = 3, 5 and 10 GeV) inside the ALEPH-TPC at LEP. We have used data simulated with two different programs giving rather ideal and realistic results, respectively. This allows a deeper inside into the functioning of the NN techniques. The results are compared with conventional methods.

In the next chapter we describe the NN learning technique we have used. Then we present two methods for the recognition of decays. In the first one every track is cut into

two halves and each half is fitted separately. Kinks show up as significant differences in the parameter sets obtained by these two fits. Since this method can also be applied in an analytical way by taking the full error matrices into account we are able to compare the efficiency of the NN algorithm directly with a conventional approach. The second method uses the big power of NN algorithms in finding correlations in the input data in a more straight-forward way by comparing the fit residuals of a full track fit.

In chapter 4 we describe the simulation of the input data, the NN architectures we have used, and the learning phase. The results are presented in chapter 5 and discussed in detail.

# 2 Learning with neural networks

For the discrimination between decaying and non-decaying tracks we use a multi-layer feed-forward network with error back-propagation [1]. In such a net the information given to the input layer is processed in several layers without feedback until it reaches the output layer. The answer of the network is compared with the correct (desired) result and the error is back-propagated from the output layer to the preceding layers. This allows to adjust the connection weights to optimize the response of the net (learning).

A typical architecture for a layered feed-forward network is shown in fig. 1. The layers between the input and the output layer are called hidden layers since one can regard such a net as a black box with defined inputs and outputs.

The input to neuron i in layer l+1 is the weighted sum of the output of all neurons in the preceding layer l plus a bias term

$$I_i^{l+1} = \sum_j W_{ij}^{l+1} \; S_j^l + B_i^{l+1} . \tag{1}$$

The sum runs over the neurons j of the preceding layer (l), $S_j^l$ is the state of the neuron j, $W_{ij}^{l+1}$ the connection weight between neuron i and neuron j, and $B_i^{l+1}$ the bias input to neuron i. This bias input is necessary to map any well behaved function between the input and the output neurons. In addition a maximum of two layers of hidden neurons is needed for neural network learning.

The state (activation) of the neurons is a function of their inputs

$$S_j^l = f(I_j^l) . \tag{2}$$

f is called activation function and we choose it in the form of the 'logistic function'

$$f(I) = \frac{1}{1 + e^{-I}} . \tag{3}$$

Therefore the activation of the neurons is limited to the range [0,1].

The neurons in the input layer are activated by the external input and the activation is propagated forward according to eq. (1) until it reaches the output layer. The activation of the output neurons is the answer of the net to the external input.

We distinguish 3 phases in the use of such learning algorithms: learning, test, and application (production).

## 2.1 Learning phase

For the learning and testing we have to know the correct answer. In the learning phase we have to adjust the connection weights. We start with random weights and we compare the output $O_i$ of the net with the correct result $D_i$. The total error in the output layer

$$E = \frac{1}{2}\sum_i (O_i - D_i)^2 \tag{4}$$

is back-propagated to the preceding layers. In order to obtain the correct output we have to modify the weights by

$$\Delta W_{ij}^{l+1} = -\epsilon \frac{\partial E}{\partial W_{ij}^{l+1}} \; . \tag{5}$$

$\epsilon$ is called learning speed since it controls the speed of the convergence process. From eqs. (1) and (2) follows using the chain rule

$$\frac{\partial E}{\partial W_{ij}^{l+1}} = \frac{\partial E}{\partial S_i^{l+1}} \frac{\partial S_i^{l+1}}{\partial W_{ij}^{l+1}} = \frac{\partial E}{\partial S_i^{l+1}} f'(I_i^{l+1}) S_j^l \; . \tag{6}$$

Differentiating eq. (3) gives

$$f'(I_i^{l+1}) = f(I_i^{l+1})[1 - f(I_i^{l+1})] \; . \tag{7}$$

In a similar way we can expand $\partial E / \partial S_i^l$ by summing over all neurons which receive input from neuron i in layer l

$$\frac{\partial E}{\partial S_i^l} = \sum_j \frac{\partial E}{\partial I_j^{l+1}} \frac{\partial I_j^{l+1}}{\partial S_i^l} = \sum_j \frac{\partial E}{\partial I_j^{l+1}} W_{ij}^{l+1} . \tag{8}$$

From

$$\frac{\partial E}{\partial I_j^{l+1}} = \frac{\partial E}{\partial S_j^{l+1}} \frac{\partial S_j^{l+1}}{\partial I_j^{l+1}} = \frac{\partial E}{\partial S_j^{l+1}} f'(I_j^{l+1}) \tag{9}$$

follows the recursive formula

$$\frac{\partial E}{\partial S_i^l} = \sum_j \frac{\partial E}{\partial S_j^{l+1}} f'(I_j^{l+1}) W_{ij}^{l+1} \tag{10}$$

which expresses the errors in layer l as a function of the following layer l+1. For the output layer we get from eq. (4)

$$\frac{\partial E}{\partial S_i^l} = O_i - D_i \tag{11}$$

and we back-propagate these errors to the preceding layers using eq. (10). The weights are then modified according to eqs. (5) and (6).

Since we can treat the input bias $B_i^{l+1}$ like a weight $W_{ij}^{l+1}$ multiplied by the activation $S_j^l = 1$ we proceed in the update of the bias terms in the same way as above by setting $S_j^l = 1$.

This procedure is repeated for all input sets. If the learning sample is not sufficiently large, which is usually the case, then we have to loop over it until we reach optimal performance on the test sample.

## 2.2 Test phase

After the weights have been adjusted in the learning step we have to check the reaction of the network if we confront it with input sets which it did not learn. In this phase we present a huge new input sample to the net and check its performance using the weights learned before.

## 2.3 Application phase

After the test has shown a satisfactory performance of the NN algorithm we can use it in a 'real' environment where we do not know the correct result. The NN runs in the same way as in the test phase.

# 3 Kink search algorithms

An important task in pattern recognition is to find out whether a well reconstructed track has a small kink due to a decay. We have to distinguish between random effects coming from statistical fluctuations in the coordinate measurements and from multiple scattering and the systematic effects caused by the decay of a track.

## 3.1 Conventional methods

A standard method to eliminate kinked tracks is to cut on the impact parameter obtained from a track fit. Although this method works well for bigger kinks it has no power for tiny kinks where the systematic effects from the decay are not significantly bigger than the statistical fluctuations.

Another way is the normal $\chi^2$ test where we attribute no kink to the tracks with a $\chi^2$ value of a normal fit below a certain cut and a kink otherwise. This method has the disadvantage that the kink hypothesis is rather weakly defined and already one badly measured coordinate can lead to a large $\chi^2$ value and thus mimic a decay. Since this test is based on the residuals of a track fit we denote it by $\chi^2_{res}$ (in distinction from other $\chi^2$ tests).

A much better method to find small kinks is to cut the track into two halves and to compare the fit parameters of both halves. The $\chi^2$ value for the hypothesis that the two parameter sets $P_1$ and $P_2$ are the same within the errors is given by

$$\chi^2_{par} = (P_1 - P_2)^T (C_1 + C_2)^{-1} (P_1 - P_2) \tag{12}$$

$C_1$ and $C_2$ being the two covariance matrices. From this $\chi^2$ value with 5 degrees of freedom we obtain directly the probability of the non-kink hypothesis. Kinks show up as a peek at low probabilities, usually between 0 and 0.01.

This method has been studied in detail for the ALEPH-TPC [2] and is implemented in the ALEPH reconstruction program JULIA.

### 3.2 Neural network with track parameters

Since learning NNs are very powerful in finding correlations between the inputs we shall use them for the comparison of the track parameters of the two halves directly and without the full error matrix calculations. We just provide as input the difference of each helix parameter between the fit of the first and the second half of a track and normalize these differences by 3 times their errors. Hence a 3 sigma deviation leads to an input value of $\pm 1$.

### 3.3 Neural network with fit residuals

We also can try to treat the kink problem in a more direct way by using the fit residuals as input of a NN. This has the disadvantage that we need a bigger network but it gives us a better understanding of the behaviour of NN algorithms in our application and it allows to evaluate the performance of the analytical parameter comparison method in an independent way. Analougously to the parameter test described above we divide the residuals by three times the error on the coordinate measurement.

This method can be applied on reduced NNs since we can average triplets of fit residuals to give one input to the NN. The fact that the residuals in the X-Y plane and in the R-Z plane are only very weakly correlated allows the use of not-fully connected NNs.

## 4 Performance test of the algorithms

### 4.1 Track simulation and selection

For our study we have used simulated data of the ALEPH-TPC. The Time Projection Chamber of the ALEPH experiment at the LEP collider at CERN is a cylindrical drift chamber providing three-dimensional track coordinates at 21 radial layers [3]. Since the TPC is inside an axial magnetic field the trajectory of a charged particle is a helix. Its projection onto the X-Y plane is an arc of a circle and onto the R-Z plane ($R = \sqrt{X^2 + Y^2}$) almost a straight line.

For the decay samples we have choosen the decays $\pi^{\pm} \rightarrow \mu^{\pm} + \nu$ with $\pi$ energies of 3, 5 and 10 GeV which cover the two most interesting energy regions. We need for the learning also a sample of non-decaying tracks. We have taken primary muons with the same energies.

At 3 GeV the maximal decay angle of about 13 mrad is already so big that we find kinks with high efficiency (about 80% ). Here we can test the performance near saturation and the influence of multiple scattering. At 10 GeV we can ignore multiple scattering but the maximal decay angle is much smaller (about 4 mrad). This means that the systematic shifts of the coordinates due to the decay are comparable with the statistical fluctuations in the measurement of the coordinates. Thus the kink finding efficiency is only about 40% at 10 GeV. We consider therefore this energy as the real test of the performance of the algorithms.

Such single track events were simulated with the standard Aleph Monte Carlo program GALEPH and the pions where forced to decay well inside the TPC (at least 15 cm away from the edges of the TPC, which insures that each of the two tracks has at least 3 coordinates). The direction (polar angle) of the tracks was choosen such that the tracks crossed the whole TPC in radial direction. In order to study the dependence of the results on the details of the simulation we have used two different simulation levels. A fast simulation (FSIM) which gives rather ideal coordinates and errors (running GALEPH without the option TPCSIM) and a detailed simulation (DSIM) which leads to realistic $\chi^2$ distributions (running GALEPH with the option TPCSIM).

The simulated data were reconstructed with the Aleph reconstruction program JULIA.

## 4.2 Network architectures

### 4.2.1 Parameter method

We want to train neural nets by providing the difference of each of the 5 helix parameters normalized by the error as input and by requiering the output 0 for non-decaying tracks and 1 for decaying tracks. This means that we need neural networks with 5 input and 1 output neuron and we have to find the best configuration of hidden neurons. We have found good performance with 1 hidden layer containing 5 and 10 neurons, respectively. These architectures are summarized in the first part of table 1 (the residual input is explained in the next subsection).

| input | no. of neurons per layer | comments |
|---|---|---|
| parameters | $5-5-1$ | |
| | $5-10-1$ | |
| residuals | $14-7-1$ | averaged residuals |
| | $14-14-1$ | averaged residuals |
| | $42-6-1$ | layer 1 and 2 'half' connected |

TABLE 1
Principle NN architectures used in this study.

The connection between the layers is complete. Each neuron is connected to all neurons in the following layer (see fig. 1). The efficiencies of the two layouts at $\pi$ energies of 3, 5 and 10 GeV are given in table 2a for FSIM data and in table 2b for DSIM data.

| layout | 3 $GeV$ | 5 $GeV$ | 10 $GeV$ |
|---|---|---|---|
| $5-5-1$ | 79.7 | 69.8 | 54.6 |
| $5-10-1$ | 77.1 | 68.0 | 53.6 |
| $14-7-1$ | 80.7 | 67.3 | 52.5 |
| $14-14-1$ | 81.1 | 67.4 | 52.8 |
| $42-6-1$ | 80.3 | 68.9 | 55.4 |

TABLE 2a
Dependence of the kink finding efficiencies [%] on the NN architecture for FSIM data.

| layout | 3 $GeV$ | 5 $GeV$ | 10 $GeV$ |
|---|---|---|---|
| $5-5-1$ | 78.9 | 67.0 | 53.5 |
| $5-10-1$ | 79.1 | 67.2 | 53.6 |
| $14-7-1$ | 79.9 | 65.5 | 51.5 |
| $14-14-1$ | 80.5 | 65.7 | 53.9 |
| $42-6-1$ | 80.3 | 67.7 | 54.5 |

TABLE 2b
Dependence of the kink finding efficiencies [%] on the NN architecture for DSIM data.

Since the differences between both NNs are very small we have decided to use the simpler network for a more detailed study. We denote this 5-5-1 network in the following by $N_{par}$.

### 4.2.2 Residual method

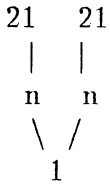
Since the number of residuals is usually much higher than the number of parameters we are confronted here with two problems: a much bigger learning effort and memorization instead of learning. If the number of different training input sets is not sufficently bigger than the number of connections between the neurons then the NN does not really learn the basic features of the problem but it rather memorizes the training sample. This means that the NN becomes able to react well to the training data but it can not generalize. Generalization is the property of a trained NN to behave well for data which it did not learn.

In order to keep the CPU time consumption for the simulation of the input data and the learning phase reasonable we try to cut down the size of the NN without loosing efficiency. One way to do that is input reduction. Since we are looking for systematic effects we can average several residuals to give one NN input value. We have found the best performance by averaging 3 residuals. Since we have 21 residuals per track in the two plans we reduce the number of inputs from 42 to 14. We studied extensively NNs with 7 and 14 hidden neurons in one layer.

Another method is to use the full input but to tailor the NN. The fact that the X-Y and the R-Z residuals are only very weakly correlated allows us to treat the X-Y and

the R-Z information separately in the hidden layer. By this we mean that we split the hidden layer into two halves. The neurons in the first half are connected with all X-Y residuals without any connection with R-Z residuals. The second half is fully connected with the R-Z inputs but not with X-Y inputs. The output neuron is connected with all hidden neurons and combines therefore the information from the two planes. This layout is shown in fig. 2.

Since we have 21 inputs for each plane and 1 output neuron we can describe such layouts in the following way




where n denotes the number of hidden neurons in each half. We have tested layouts with n = 21, 7, 3, and 2. The first three networks give the same results whereas the efficiency becomes rather poor for n=2. We therefore have chosen n=3 and write its layout in the form 42-6-1. In order to check the performance of these 'reduced' nets we studied the fully connected NN architectures 42-21-1 and 42-42-1. Since their results are slightly worse we do not consider them any longer.

The performance of the NNs is summarized in table 2a and 2b. Since the best results are obtained with the 'half' connected 42-6-1 layout we shall concentrate on it and call it $N_{res}$ in distinction from $N_{par}$, our favoured network for the parameter method.

### 4.2.3 Networks with 2 hidden layers

We also have studied NNs with two hidden layers. For the parameter method we obtained with the layouts 5-5-5-1 and 5-10-10-1 the same results as with 5-5-1. For the residual approach we tested the fully connected configuration 42-42-21-1 which showed about the same efficiency as the 42-21-1 and 42-42-1 layouts. From this we conclude that we do not gain by using two hidden layers and we shall concentrate on the most promissing NNs with one hidden layer.

## 4.3 Learning phase

For the learning we have used about 40000 tracks for each energy and presented them to the networks many times (between 50 and 150 times depending on the size of the network) in random order. This randomness avoids cycling effects (the networks learns to adjust to the order of the input events) and makes the convergence faster. Since the generation of tracks is rather time consuming (more than 2 sec per DSIM track on an IBM 3090) we have to loop over the input tracks and we can not afford to provide for each learning step a new input. In order to decrease the number of spurious kinks (non-decaying tracks which get a kink assigned) we have used 2 times more non-decaying

tracks than decaying tracks. The learning speed $\epsilon$ was set to 1 at the beginning and decreased to 0.1 towards the end of the iterations.

The learning takes roughly between 10 and 20 minutes (in IBM-3090 units) depending on the NN layout and is therefore considerably faster than the track generation.

# 5 Results

For the discrimination between the two hypotheses kink and non-kink we have choosen for each network architecture the cut in the output distribution such that the percentage of spurious kinks is constant. This cut was set to 1% for FSIM tracks and to 5% for DSIM tracks since the number of non-decaying tracks with bad fits is much bigger for DSIM data.

The efficiency of the neural networks can directly be compared to the results of the analytical parameter comparison method.

In order to test the algorithms under optimal conditions we select for the following tests only tracks without missing coordinates (i.e. tracks with 21 coordinates reconstructed by JULIA). The effect of relaxing this condition will be discussed in subsection 5.5.

## 5.1 Results for mixed charges

We tested the best NN layouts for the parameter ($N_{par}$) and the residual method ($N_{res}$) with learning samples and test samples containing 50% positive and 50% negative charged tracks. The results are summarized in table 3a and 3b which contain also the efficiencies of the conventional tests $\chi^2_{res}$ and $\chi^2_{par}$ (see section 3.1).

| $E_\pi$ | 3 GeV | 5 GeV | 10 GeV |
|---|---|---|---|
| $\chi^2_{res}$ | 73.5 | 57.8 | 38.0 |
| $\chi^2_{par}$ | 79.8 | 68.0 | 50.2 |
| $N_{res}$ | 75.2 | 68.5 | 50.1 |
| $N_{par}$ | 79.0 | 69.1 | 51.6 |

TABLE 3a
Kink finding efficiencies [%] for FSIM data.

| $E_\pi$ | 3 GeV | 5 GeV | 10 GeV |
|---|---|---|---|
| $\chi^2_{res}$ | 53.1 | 35.8 | 14.3 |
| $\chi^2_{par}$ | 76.0 | 62.0 | 40.2 |
| $N_{res}$ | 79.3 | 65.3 | 48.1 |
| $N_{par}$ | 78.3 | 65.9 | 47.0 |

TABLE 3b
Kink finding efficiencies [%] for DSIM data.

It is evident that the second test is much more efficent and we shall use it to evaluate the performance of the NN methods.

The efficiencies of both NN approaches are rather similar except for the case of FSIM data at 3 GeV. The poorer performance of the residual method can be attributed to multiple scattering effects which disturbe the rather ideal (FSIM data) residuals more than the parameters which are obtained by a fit over about 10 coordinates.

Furthermore the NN efficiencies are comparable with the $\chi^2_{par}$ method for FSIM data, but substantially better for DSIM data. This can easily be understood in the following way: for ideally simulated data the errors on the coordinates are determined correctly which leads to correct errors on the fit parameters. The analytical $\chi^2_{par}$ method gives therefore optimal results and the NN algorithms can not do better (except from $N_{par}$ at 10 GeV, which is discussed below). We can consider the agreement between $N_{par}$ and $\chi^2_{par}$ as proof for the good functioning of the NN algorithm. This means that the NN has 'learned' correctly the correlations between the two parameter sets of the track fits. For DSIM data the analytical method suffers much more from wrong coordinate errors than the NN approach which is apparently able to adapt to systematic effects.

A nice example for the power of NN methods to find correlations in the input data is the above mentioned difference between $N_{par}(51.6\%)$ and $\chi^2_{par}(50.2\%)$ for FSIM at 10 GeV in table 3a. Although the difference does not seem to be significant we looked into this case in more detail. A comparison of the connection weights of the network shows that the NN has learned to ignore 2 helix parameters almost completely. These two parameters are the dip angle and $Z_0$ (the Z coordinate of the helix at the point of closest approach to the origin) which are determined by the coordinates in the R-Z plane with a negligible contribution from the X-Y plane. The resolution of the ALEPH-TPC in R-Z is almost an order of magnitude worse than in X-Y and for 10 GeV tracks not sufficent to detect any systematic effect in this plane. The NN has therefore learned correctly to ignore these two parameters. In the analytical method we add to the contribution from the three relevant parameters some sort of random noise from the two others which diminishes the discrimination power of this method. Using only the three discriminant parameters in the $\chi^2_{par}$ test improves its performance by about 2% (in good agreement with $N_{par}$). No difference was found for DSIM data which shows that the 'weak' parameters provide some protection against badly measured coordinates.

## 5.2 Results with charge dependence

We want to check now whether the charge of the tracks (i.e. the direction of the bending in the X-Y plane due to the magnetic field) has some detectable systematic effects. The ability to find small kinks depends for example strongly on the direction of the decay in the X-Y plane since the secondary track (the $\mu$ in our case) has less energy and bends therefore more in the magnetic field. If the $\pi$ decays in the X-Y plane towards the origin then we have two additive effects which shift the coordinates of the generated $\mu$ compared to the extrapolation of the $\pi$ into the same direction: the change of the direction (decay angle) and the change of the curvature. If the $\pi$ decays into the opposite direction then both effects interfere and it becomes much harder to detect the decay.

In order to study such effects quantitively we learn with tracks of one charge and we test with tracks of the opposite charge by changing the sign of the parameter inputs and the X-Y residual inputs. The results are summarized in table 4a and 4b.

| $E_\pi$ | 3 $GeV$ | 5 $GeV$ | 10 $GeV$ |
|---|---|---|---|
| $\chi^2_{par}$ | 79.8 | 68.0 | 50.2 |
| $N_{res}$ | 80.3 | 68.9 | 55.4 |
| $N_{par}$ | 79.7 | 69.8 | 54.6 |

TABLE 4a
Kink finding efficiencies [%] with charge dependence for FSIM data.

| $E_\pi$ | 3 $GeV$ | 5 $GeV$ | 10 $GeV$ |
|---|---|---|---|
| $\chi^2_{par}$ | 76.0 | 62.0 | 40.2 |
| $N_{res}$ | 80.3 | 67.7 | 54.5 |
| $N_{par}$ | 78.9 | 67.0 | 53.5 |

TABLE 4b
Kink finding efficiencies [%] with charge dependence for DSIM data.

Comparing with table 3a and 3b we see that the effect is rather big. The performance of the NN methods for the hard test case (DSIM data at 10 GeV) is excellent. 54.5% and 53.5% efficiency of the NN residual and parameter test, respectively, compared to 40.2% of the conventional $\chi^2_{par}$ method.

## 5.3 Dependence on simulation details

One big problem frequently encountered in learning methods is the strong dependence of their performance on features in the input data which are not relevant for the problem. To study the robustness of our NN algorithms against such perturbations we

apply the following cross tests: we learn with FSIM data and test with DSIM data and the other way round, taking in both cases the charge of the tracks into account. The results are listed in table 5a and 5b.

| $E_\pi$ | 3 GeV | 5 GeV | 10 GeV |
|---|---|---|---|
| $\chi^2_{par}$ | 79.8 | 68.0 | 50.2 |
| $N_{res}$ | 74.9 | 69.7 | 54.4 |
| $N_{par}$ | 77.9 | 67.1 | 54.1 |

TABLE 5a
Kink finding efficiencies [%] with charge dependence for FSIM data after learning on DSIM data.

| $E_\pi$ | 3 GeV | 5 GeV | 10 GeV |
|---|---|---|---|
| $\chi^2_{par}$ | 76.0 | 62.0 | 40.2 |
| $N_{res}$ | 80.8 | 66.3 | 54.1 |
| $N_{par}$ | 78.8 | 66.3 | 51.6 |

TABLE 5b
Kink finding efficiencies [%] with charge dependence for DSIM data after learning on FSIM data.

We observe a very small dependence on the quality of the input data since the difference between the two simulation programs is rather big. This gives us good confidence that one can apply such techniques with high efficiency to real data after training on well simulated data.

## 5.4 Energy dependence

Finally we want to show the dependence of the learning on the energy. Since our samples cover two extremes, rather easily detectable decays with perturbations from multiple scattering at 3 GeV and tiny but clean decays at 10 GeV, we can not expect to find one single set of connection weights which performs well for both energies. Nevertheless we check this dependence directly by applying the weights learned at each of the 3 energies to test data for the two other energies. The results are presented in table 6a and 6b.

| method | $E_\pi^{test}$ | $E_\pi^{learn}$ | | |
|---|---|---|---|---|
| | | 3 GeV | 5 GeV | 10 GeV |
| $N_{res}$ | 3 GeV | 80.3 | 73.4 | 71.3 |
| $N_{par}$ | | 79.7 | 79.6 | 74.8 |
| $N_{res}$ | 5 GeV | 70.2 | 68.9 | 68.8 |
| $N_{par}$ | | 67.2 | 69.8 | 68.1 |
| $N_{res}$ | 10 GeV | 45.6 | 53.5 | 55.4 |
| $N_{par}$ | | 46.5 | 51.9 | 54.6 |

TABLE 6a
Energy dependence of the kink finding efficiencies [%] with charge dependence for FSIM data (learning ($E_\pi^{learn}$) and testing ($E_\pi^{test}$) at different energies).

| method | $E_\pi^{test}$ | $E_\pi^{learn}$ | | |
|---|---|---|---|---|
| | | 3 GeV | 5 GeV | 10 GeV |
| $N_{res}$ | 3 GeV | 80.3 | 75.6 | 75.8 |
| $N_{par}$ | | 78.9 | 77.9 | 76.2 |
| $N_{res}$ | 5 GeV | 67.7 | 67.7 | 68.6 |
| $N_{par}$ | | 64.5 | 67.0 | 67.8 |
| $N_{res}$ | 10 GeV | 46.2 | 50.2 | 54.5 |
| $N_{par}$ | | 43.4 | 49.4 | 53.5 |

TABLE 6b
Energy dependence of the kink finding efficiencies [%] with charge dependence for DSIM data (learning ($E_\pi^{learn}$) and testing ($E_\pi^{test}$) at different energies).

The diagonal terms are taken from table 4a and 4b, respectively, and serve as reference for the optimal value at each energy. Since the energy dependence is rather smooth we can devide the whole energy spectrum into a small number of energy regions for which we have to learn the weights separately.

## 5.5 Missing coordinates

As we have seen so far both NN techniques work almost equally well for completely measured tracks. In real events we have to cope with the fact that often not all track coordinates can be measured because of dead regions in the detector, overlapping tracks, etc. Since these problems are very detector dependent we did not study them in detail. Nevertheless we can make some rather general remarks.

At first glance the NN parameter method seems to be better suited to handle such tracks since it does not directly depend on the number of coordinates. But this technique suffers apparently from the fact that the correlation between the parameters can not be learned sufficiently well. The NN algorithm is therefore slightly less efficient

than the analytical method if we do not specially learn different configurations of missing coordinates which might be a rather long task.

Although the NN layout for the residual method depends on the number of measured track coordinates we found a very statisfactory behaviour of this technique without additional learning. We train the network as before on completely measured tracks and we set in the test the activation of the input neurons which correspond to missing coordinates to 0. The improvement of the efficiency of this method compared to the conventional one is about the same as described above. The results are summarized in table 7a and 7b.

| $E_\pi$ | 3 $GeV$ | 5 $GeV$ | 10 $GeV$ |
|---|---|---|---|
| $\chi^2_{par}$ | 78.0 | 65.5 | 46.4 |
| $N_{res}$ | 78.3 | 66.9 | 52.5 |
| $N_{par}$ | 75.9 | 65.1 | 50.0 |

TABLE 7a

Kink finding efficiencies [%] with charge dependence for FSIM data with missing coordinates.

| $E_\pi$ | 3 $GeV$ | 5 $GeV$ | 10 $GeV$ |
|---|---|---|---|
| $\chi^2_{par}$ | 73.4 | 58.2 | 36.1 |
| $N_{res}$ | 76.8 | 63.4 | 49.9 |
| $N_{par}$ | 74.4 | 62.6 | 47.2 |

TABLE 7b

Kink finding efficiencies [%] with charge dependence for DSIM data with missing coordinates.

## 5.6 Execution time

Because of the rather small size of the NN layouts the time needed to calculate the answer (output) of the network is small (less than 0.1 msec on an IBM 3090) compared to a helix fit (about 0.6 msec). The analytical parameter comparison method, which demands two additional fits and one matrix inversion, is therefore about 20 times slower than the NN residual method if one assumes to get the residuals as by-product of the normal track fit. The NN parameter approach is like the analytical method limited by the time needed to fit both track halves.

# 6 Conclusion

We have studied extensively several possibilities to use neural network techniques for the recognition of decays of charged tracks. We have found excellent performance and robustness for two different methods using as input track fit parameters and residuals, respectively. Since the residual approach is faster and easier to handle for tracks with missing coordinates we prefer it over the other.

The NN residual method is more efficient and about 20 times faster than a conventional kink-search algorithm based on the comparison of the track parameters obtained from two half-track fits. In the case of $\pi^{\pm} \rightarrow \mu^{\pm} + \nu$ with $E_{\pi} = 10GeV$ 54.5% of the decays are detected with this NN technique compared to 40.2% with the conventional method.

Since these NN layouts have a rather small number of neurons and connections no special hardware is needed for an efficient implementation.

# Acknowledgment

I would like to thank Lluís Garrido for many stimulating discussions.

# References

[1] D.E. Rumelhart et al., Nature 323 (1986) 533.

[2] G. Stimpfl-Abele, internal note ALEPH 89-119 (1989).

[3] The Aleph Collaboration, Nucl. Instrum. Methods A 294 (1990) 121.

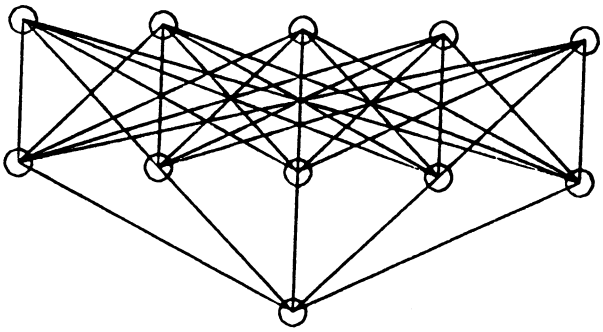


**Fig. 1:** Schematic view of a normal layered feed-forward network with one hidden layer (the neurons are represented by circles and the connections by lines).

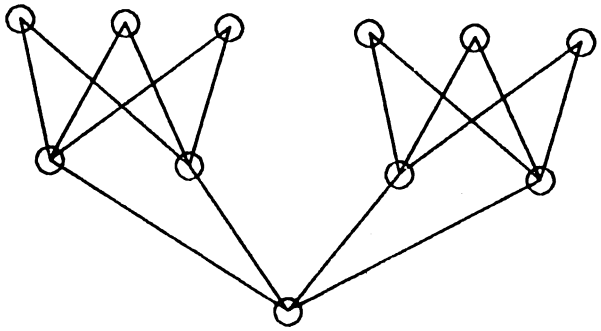


**Fig. 2:** Schematic view of the half-connected feed-forward network used for the residual method (the neurons are represented by circles and the connections by lines).